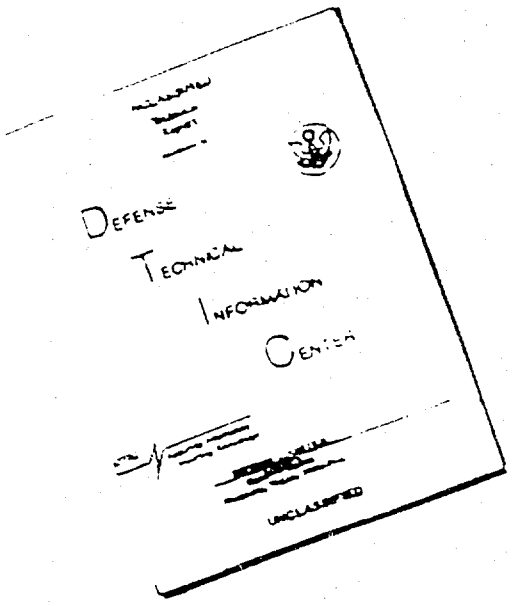
Contract No. DAAG-53-76-C-0043
U. S. Army Mobility Equipment Research
and Development Center,
Fort Belvoir, Va. 22060

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

(12)

A COMPUTER SIMULATION SYSTEM FOR THE PRODUCTION OF REALISTIC PHOTOGRAPHS OF TERRESTRIAL SCENES

Contract No DAAG 53-76-C-0043

P-7126

A COMPUTER SIMULATION SYSTEM FOR THE PRODUCTION OF REALISTIC PHOTOGRAPHS OF TERRESTRIAL SCENES

by

Joan Brooks
Robert Goldstein
Herbert A. Steinberg

Final Report

July 1976

Contract No. DAAG53-76-C-0043
U. S. Army Mobility Equipment Research and Development Center
Fort Belvoir, Va. 22060

Mathematical Applications Group, Inc.
3 Westchester Plaza
Elmsford, New York 10523

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This work represents the culmination of efforts supported by Ballistic Research Laboratories and Mobility Equipment Research and Development Command to develop a computer simulation system for large scale terrestrial scenes. The previous efforts have produced independent models for solid targets, camouflage nets, terrain and vegetation ground cover. The present effort has produced automatic generation of vegetation input, improvements in the efficiency of the vegetation model and increased flexibility of the terrain model and has welded the independent models into a single system capable of producing shadowed photographs of all or any part of a composite scene. The system has been successfully demonstrated by the production of a high quality photograph of a simulated test site.

1. INTRODUCTION AND OVERVIEW OF THE SYSTEM

1.1 Background

The computer system described in this report represents the successful culmination of efforts of Mathematical Applications Group, Incorporated, supported by Ballistic Research Laboratories and Mobility Equipment Research and Development Command and others, to develop a practical tool for realistic computer simulation of large scale terrestrial scenes.

The previous efforts have produced independent models for solid targets[1], camouflage nets[2], terrain[3] and vegetation ground cover[3,4]. Although independent, these models were mutually consistent in that they produced images in the same format (block format image) and could, therefore, be merged into a composite scene. Associated with these models were a group of utilities for merging, magnifying and shifting of images and a conversion program to produce from the block format the string format required for the final phases of picture making.

From a practical point of view, there were a number of deficiencies in some of the models and in the system as a whole. Specifically:

a) The ray tracing elements of the vegetation model required too much computer time. This was, in part, a result of the multi-stage ray tracing and, in part, a result of the great complexity required for close-up photography, but wasted on distant trees. Thus, both a more efficient ray tracing was suggested and a different, less complex, model was needed for the low resolution situation.

b) The generation of "forest" data, i.e., locations, sizes and orientations of the trees of a scene, was performed by hand and this placed a literally impossible burden on the user.

c) The terrain model was a modification of the camouflage net which tied the color and texture boundaries to the resident net patterns. This gave little flexibility in the description of terrain. Furthermore, the input was precisely the input for a camouflage net, and thus did not proceed directly from the field data available to modelers of terrain.

d) The generation of a composite photograph required many separate problem set-ups and submissions: one for each model and for its' shadow and, for the vegetation model, separate runs for each tree type. The calendar time for generation of such a photograph might be many days.

All of these problems have been addressed successfully and the solutions are described in detail in subsequent sections of this report. In this section, we will outline the improvements and guide the reader to the more detailed descriptions.

1.2 More Efficient Ray Tracing for the Vegetation Model

A feature of both the construction and the ray tracing of the vegetation model was the multi-stage, prototype concept. In essence, this means that the tree is built up in stages, using copies of a prototype secondary branch to construct a primary branch prototype, and copies of the primary branch prototype to construct the framework of the tree. In ray tracing, the process is reversed: when a volume containing a copy is struck by a ray, the ray is transformed, i.e., translated and reoriented, into the (possibly) magnified world of the prototype for detailed ray tracing within that structure. Because the volumes occupied by the copies inevitably overlap a good deal, this procedure tends to invoke much more ray tracing than is required for the more open structure of a one stage model.

Thus, the essential feature of the new models is the elimination of some of the multi-staging in the ray tracing. The construction of the tree proceeds as before (from the same input data), but only the digitized leaf and twig structure is represented by copies of BOXes in the system of the tree. All woody structure - secondary branch base, primary branch framework, and stem - is represented by truncated right cones in the tree system. A three dimensional lattice covering the containing box of the tree permits analytic, ordered selection of bodies for investigation along the ray. This reduction of the role of multi-staging has resulted in a computer time saving of a factor of four or five.

The two new models that implement the new ray tracing are referred to as the high resolution model and the low resolution model. The high resolution model is geometrically identical with the old model. The low resolution model differs from the high resolution model in that the BOXes containing leaf and twig structure are probed "statistically", i.e., penetration of a struck BOX is probabilistic, as is the decision whether leaf or twig has been struck. Thus, this model differs from the others both geometrically and in its ray tracing method. Because another "stage" has been removed, the ray tracing is roughly twice as fast as for the high resolution model. The statistical data for this model is derived from ray tracing "experiments" on the high resolution model.

These two models were implemented by the government and applied to four tree types. The resulting eight data sets constitute the "tree library" described in Section 2. The details of the construction technique are given in Appendix A.

1.3 The Data Base

The data required to describe any given test site consists of five libraries. Two of these are related to vegetation ground cover and are 1) the collection of high and low resolution models for each of the tree types of the region and 2) the forest library. Other libraries are the terrain library, the Combinatorial Geometry target library and the camouflage net library.

Since a prime objective of this effort was to reduce the user burden, it was decided that these libraries should exist on a single logical unit in a highly processed state and be accessed by "knowledgeable" retrieval routines, i.e., routines which are aware of scene limits without unnecessary user prompting. The degree of processing in the stored data was determined for each model individually. For new models (vegetation and terrain) the stored data are the fully processed contents of arrays and common blocks, ready for ray tracing, while for the forest description, the twelve parameters of the containing BOX for each tree are stored. The net and CG target data are, however, stored as

required by these working models: card images for the CG and a mix of card images and unformatted restart data for the nets.

The format of this tape and the several retrieval methods is discussed in detail in Section 2. Data selection from the forest library and data organizing for the vegetation ray tracing will also be discussed there.

1.4 The Forest Generator

A stand-alone program for the generation of the forest library has been written and utilized in this effort. This program uses resident distribution functions with user-specified means and variances to distribute trees of given types within given boundaries or along given contours. Among the properties selected from distribution functions are location, orientation, height and cluster size.

This program is fully described in Section 3, which also contains a user's guide and input description.

1.5 The Terrain Model

Modifications of the terrain model have proceeded by separating the "installation" procedure from the ray tracing procedure, and building a stand-alone program to produce the processed data required for the data base. This stand-alone program requires elevation data, arbitrary color boundaries in a horizontal plane and associated region number assignments. Road data, as in the old model, are also required. Only minor modifications were required in the ray-tracing to identify the color boundary within which a strike occurs.

The installation program and the modifications in the ray tracing are fully described in Section 4. That section also contains a user's guide and input description for the installation program.

1.6 The Image Generating System

The image generating elements and shadow generating elements of the four basic geometry models have been integrated into a system which, operating from the data base, can produce all or any part of a scene as a block format image (for further processing) or as a string format image, ready for the color pass of the MAGI SYNTHAVISION process. The other components of the system are the "scene setting" components, for reading and processing camera data and sun data, the data retrieval components for selecting appropriate data from the data base, the MERGE program for merging two block format images, and the CONV program, for converting the block format image to the string format required by the color pass.

All of these components are under the control of an executive (PROGRAM TRAFFIC) whose function is to read and respond to option cards and thus direct the flow of a user initiated sequence of runs. All of the system components mentioned above are opted by the user, with the single exception of the data retrieval option, for which the executive takes the responsibility. The user also names the input and output files for a run and thus decides whether a current image is to be merged with a previously produced image as the new image is generated or whether images are to be saved separately for later merges.

The only card input required by the system are the option cards (two for most options) and the scene-setting data: camera data, sun data, translation and scale data for the targets and nets, raster limits for each geometry and shadow run, and color assignment cards for the conversion run. For a typical whole scene sequence with one target and one net, there will be four geometry runs, four shadow runs, two data runs and one conversion run. Apart from the color assignment cards of the conversion run, fewer than fifty cards are required.

Because of the size and complexity of the system, it has been necessary to structure it in overlays. Since the overlay method is machine-dependent, the IBM version used at MAGI in the production phase of this work is different from that required by to the government for the CDC 6600 computer. The CDC version is the subject of Section 5, where the executive and new routines of the system are described. That section also contains a user's guide and input description.

1.7 Demonstration of the System

The completed system was debugged on a data base consisting of a fifteen-tree forest, a tree library with two tree types, a terrain based on nine elevation points, two combinatorial geometry targets and a single camouflage net.

A full scale problem, based on a test site description provided by the government, was then executed with excellent results. The details of that problem are described in Section 6 and the resulting photograph is shown. In that section we also give execution times for the data-producing codes as well as the image generating system, and make recommendations for the manner in which test-site descriptions should be provided in the future for good simulation.

2. THE DATA BASE.

2.1 Description of the Data Base

The data base for the system resides on a tape or disk file and contains all of the information about a single test site as well as all of the data for combinatorial geometry targets and camouflage nets that may be used in connection with the site.

The separate libraries or data units which comprise the data base are, in the order of occurrence on the tape:

a) Forest library

b) Tree library

c) Terrain library

d) Combinatorial Geometry target library

e) Camouflage Net library

The detailed format of the data base is given in Table 2.1. Further details of the forest library may be found in Section 3.5 of this report, while the terrain library is fully discussed in Section 4.2.3.

The forest library is divorced from the terrain library in the sense that the vertical location of the tree base is at sea level. The actual elevation is extracted from the terrain library at the time of retrieval of the tree from the forest library. For shadow runs, the normal to the terrain is also computed at this time.

Each Combinatorial Geometry target is represented by card images of the geometry deck, i.e., raw data prepared by the user. These cards are fully described in references (1) and (3). A separator record consisting of the Hollerith information 12X,FINI folows the last card image of each deck except the last, which is followed by the record containing the information 12X,STOP.

Each camouflage net requires three data sets: 1) a variable number of card images, 2) 7 records of processed binary data (the output of CAMNET on TAPE8, for restart), and 3) two final card images. The first two sets are separated by a record containing the Hollerith information 12X,SWCH. The net library, and therefore, the data base, is closed by an end of file.

The tree library describes each of several tree types by means of two models: a high resolution model and a statistical, or low resolution model. The high resolution model is geometrically identical to that described in (1) and (4) but much of the multi-staging has been removed to increase the efficiency of ray-tracing. The low resolution model is based on the high resolution model with the digitized leaf and twig structure replaced by a statistical leaf cloud. These models were prepared by the government and are described in Appendix A of this report.

The identification of a model is based on its tree type, $I(\leq 6)$, and its resolution. Thus the identifier for the high resolution model of tree type I is

$$LTYP(2*I-1) = 10*I + 1 \quad ;$$

while that for the low resolution model of the same tree type is

$$LTYP(2*I) = 10*I + 2 \quad .$$

This identifier is the first record of a tree model. The second record of the model carries the contents of the labelled common TREE of the vegetation code and consists of construction data. A third record carries some statistical data (meaningful only for the low resolution model), namely the arrays STTT(4), AVE(3) and DAVE(2), and the contents of the common blocks FG, KD,KLOUD, RAYTR and COLR of that model. There are 21631 words of data in the second record and 1680 words in the third record.

There are currently four tree types represented on the data base: a coniferous tree (LTYP(1)=11, LTYP(2)=12), two deciduous trees with leaves, the pin oak (LTYP(3)=21, LTYP(4)=22) and the cottonwood (LTYP(7)=41, LTYP(8)=42), and, finally, a deciduous tree without leaves (LTYP(5)=31, LTYP(6)=32). Room has been allowed in the retrieval routines for two more tree types.

TABLE 2.1 - FORMAT OF THE DATA BASE

DATA UNIT	RECORD NOS.	CONTENTS	COMMENTS
1	1,N	FOREST LIBRARY	Output of FORGEN
	N+1	78130	Flag
2		TREE LIBRARY	3 records for each of N models. N≤12
	1	LTYP(1)	Tree ID
	2	MA,FPD,....	
	3	STTT,AVE...	
	4	LTYP(2)	
		.	
		.	
		.	
	3*N-2	LTYP(N)	Tree ID
	3*N-1	MA,FPD ..	
	3*N	STTT,AVE..	
	3N+1	15626	Flag
3	1-9	TERRAIN LIBRARY	Output of INSTAL
	10	31252	Flag
4		CG TARGET LIBRARY	NMAX targets
	1,N1	Card images of standard input	HOLLERITH (20A4)
	N1+1	12X,FINIS	Separator
	(N1+2),(N1+N2+2)	Card images	
	N1+N2+3	12X,FINIS	Separator
	.	.	
	.	.	
	.	.	
	NMAX $\sum_{2=1}$(N2+1)	12X,STOP	Flag
5		NET LIBRARY	LMAX nets
	1,L1	L1 card images for restart option #5	HOLLERITH (20A4)
	L1+1	12X,SWCH	Flag, switch to unformatted data
	L1+2,L1+8	Processed data	7 binary records from CAMNET restart tape)
	L1+9,L1+10	2 card images	Final cards for restart
			Repeat above for each net.
	EOF		End data base.

2.2 Data Retrieval Methods and Routines

The data retrieval methods are peculiar to the library concerned. Retrieval for the vegetation modules is rather complicated, while that for the other modules is essentially a matter of tape positioning.

The retrieval routines are in a primary overlay of the system, OVERLAY(2), with main program PROGRAM DATRET. This overlay is called by the executive whenever an image generating or shadowing run is opted by the user. The geometry model concerned is identified by a program number, JSAVE, which is passed to DATRET through common. DATRET calls the appropriate subroutine:

SUBROUTINE TERRET

Called from DATRET for

1) terrain image or shadow, TERRET searches the data base for the integer flag 15626 and leaves the tape positioned for reading by input elements of the terrain modules;
2) vegetation data processing, TERRET finds the terrain library and reads into memory the first four records (see use under FORRET).

a) Called from: DATRET

b) Subroutines called: None

SUBROUTINE TARGET(N,M)

Called from DATRET for either CG targets (M=1) or camouflage nets (M=2), TARGET searches the data base for the appropriate flag and positions the file at the Nth target of either type. The data are later read by the input routines of the appropriate model. The ordinal number N is supplied by the user as described in Section 5.

a) Called from: DATRET

b) Subroutines called: None

SUBROUTINE FORRET

This subroutine is called from DATRET to retrieve data for a particular model of a particular tree type. If the model is not in the tree library or is not represented in the forest library, a flag is returned to the executive inhibiting the geometry run and calling for the next tree model.

The action of FORRET is to locate the model on the data base and copy it to a temporary storage device (the user designated logical unit IPD). FORRET then searches the forest library for trees of the correct type which are within the scene limits. Each such tree must satisfy a range criterion for the particular model (high or low resolution). To calculate the range and to complete the tree description, the vertical position of its base is extracted from the terrain library by a call to the subroutine ZANDW. This subroutine will also return the local normal to the terrain if a shadow run is contemplated. The twelve or (for shadows) fifteen word description is written on the IPD as a single record.

The range criterion for high or low resolution model is specimen-dependent. For a particular tree the criterion is that the width of the containing box shall subtend at the observation point an angle greater than 25 mils to require the use of the high resolution model. For a tree within a densely forested region, the value 50 mils is used. Such trees are tagged at the time of generation as described in Section 3.2.

a) Called from: DATRET

b) Subroutines called: ZANDW, INROUT

SUBROUTINE ZANDW

This is the routine called by FORRET to calculate the elevation, Z, of the terrain at the position (x,y) of a tree base. For shadow runs, ZANDW will also find the local normal to the terrain. For these calculations, ZANDW calls routines of the terrain ray tracing module.

a) Called from: FORRET

b) Subroutines called: INIPOS, NORM

SUBROUTINE INROUT(JFL)

This subroutine is called by FORRET to determine for a particular group of trees contained within known rectangular limits (see Section 3), whether any part of the rectangle is within the scene limits. The flag JFL is returned as non-zero if the answer is positive.

a) Called from: FORRET

b) Subroutines called: None

2.3 Forest Data Organizing Programs

Because of the very large memory requirements for a tree model (over 23000 words), it has been necessary to design the vegetation runs around a single model and to limit the number of trees in memory at one time to two hundred. A complete ground cover image is obtained by a loop on model number, with a continuous merge of the image in process with that of the previously processed models. This "merge-on-the-fly" process is described in Section 5.1. The loop extends from JFIRST (input or default, 1, to JLAST, input or default, 12).

The limit on the number of trees concurrently in memory is observed by organizing the trees of the scene by block number of the image scan (line by line, left to right within a line). This function is performed by the PROGRAM PREPAR for image generation and by the PROGRAM PREPR for shadowing by the forest.

The organized trees are written on TAPE3 in records of 200 trees, each record headed by an identifying block number. Clearly trees may be repeated for blocks of different number. A second output tape of PREPAR, TAPE10, carries for each covered block an array of dimension 2 by 20, one column of which orders, on distance from the camera, the tree numbers to be investigated for that block. The other column gives the distance.

These programs operate on the TAPE IPD, the output of FORRET, which contains those specimens of a single model which are within the scene limits established by the subroutine COVER (scene setting overlay).

3. THE FOREST GENERATOR: PROGRAM FORGEN

3.1 Introduction

The purpose of this stand-alone program is to create the data file which contains the type, size, location and orientation for every tree in the area of interest. FORGEN performs this function in response to relatively simple input data by selecting values from internal distribution functions with user-defined means and, in some cases, user-defined variances. The output data are organized by region (defined below) and, within each region, by tree type. The data for an individual tree are given as the parameters of the containing box, i.e., the vector position of one base corner and three vectors specifying the edges of the box, and therefore its size.

In the following sub-sections of this section, we will describe the philosophy and design of the program, the subroutines comprising the program, the input preparation and logical unit assignments and the format of the output file.

3.2 Program Design

The design of the program was guided by observations on the kinds of tree distributions that might be of interest and by practical considerations such as the possibility of replacing individual parts of the description or adding to the description without rerunning the entire problem.

Three kinds of tree configurations were identified as useful to efficient generation of realistic ground cover. A basic concept in the description of two of these configurations is the "cluster" and is based on the fact that, in low density distributions of trees, the unit grouping may contain more than one tree. Thus, for our purposes, trees may occur in clusters whose size is determined from an input mean and a resident distribution function. The three configurations are:

1. A uniform distribution of clusters of each of several types of trees within a closed boundary. The user defines the boundary and the types of trees and, for each tree type, the mean cluster size (up to 10), the fractional density, the mean height and the half-width of the height range.

2. A uniform distribution of clusters of each of several tree types along a "thick" line. The user defines the line, its thickness and tree parameters as in (1) above.

3. A distribution of individual trees whose types, sizes, locations and some aspects of orientation are defined by the user.

The first of these clearly lends itself to the description of forested areas or fields with scattered trees or tree clusters. The second might represent the fringe arrangement that frequently occurs near a road or on the boundaries between fields in farm country. The third type of representation is useful in the immediate neighbornood of a target, since a very special local configuration may be selected for camouflage purposes.

In what follows, we will distinguish these configurations by their geometric peculiarities and refer to them as "boundaries", "lines", and "point sets", respectively. The entire organization of the program - input, program action and output, as well as the replacement and addition of data - is based on this three-part description. The order of input, treatment and output is always: boundaries, lines, point sets.

Because the line segments comprising the boundary and the line may be determined by a road or a stream bed, the user may give a (positive) displacement for each segment such that the segment used by the codes is clear of the road or stream bed. Thus the displacement will normally be somewhat more than half the width of the bed. The direction of displacement for a boundary, whose defining points must be given counter-clockwise, is always inward. The line must be described such that the directed segment from point N to point N+1 is displaced to its left. These ideas are illustrated in Figure 3.1.

A restriction on the closed boundary should be mentioned here, namely that a line X = constant should cut the boundary in two and only two points. If a bounded region does not satisfy this restriction, it should be sectioned into two or more bounded regions.

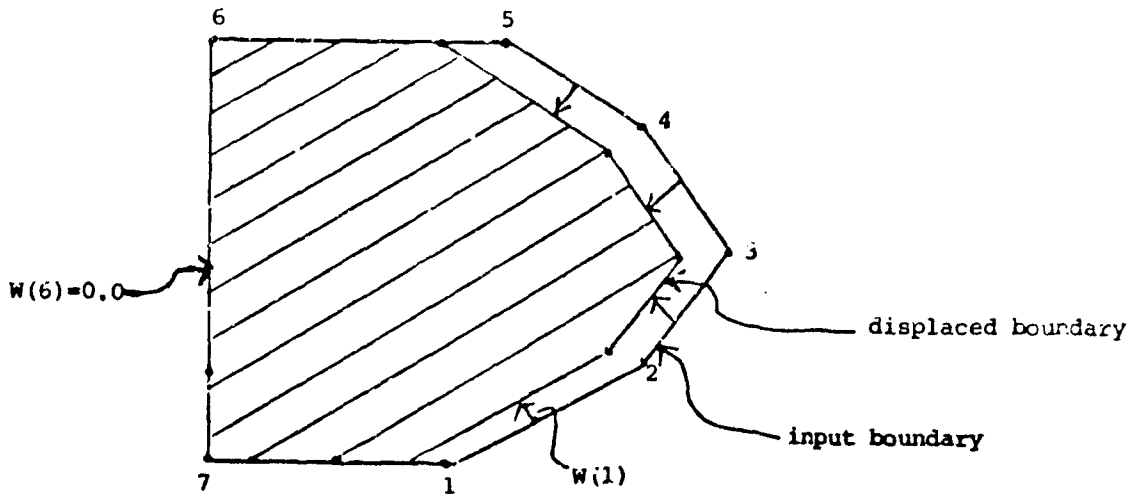
The distribution of trees within a boundary proceeds by determining the smallest containing rectangle with sides parallel to the x-y axes and covering this area with a square grid. The grid size is the product of the average width of a tree box in the area by an input factor which, in effect, determines the mean spacing of trees within a cluster. If a tree is assigned to a cell of the grid the base location will be chosen somewhere within the cell.

Once the grid has been established, an array is set up which establishes, for each value of the x index, I, the limits of the y index, J, that correspond to the boundary. Simultaneously, a pointer array is set up, such that for each value of I, the cumulative number of grid points through I-1 is recorded. These pointers refer to the "IQ" array, which will contain the identifiers for trees assigned to grid points.

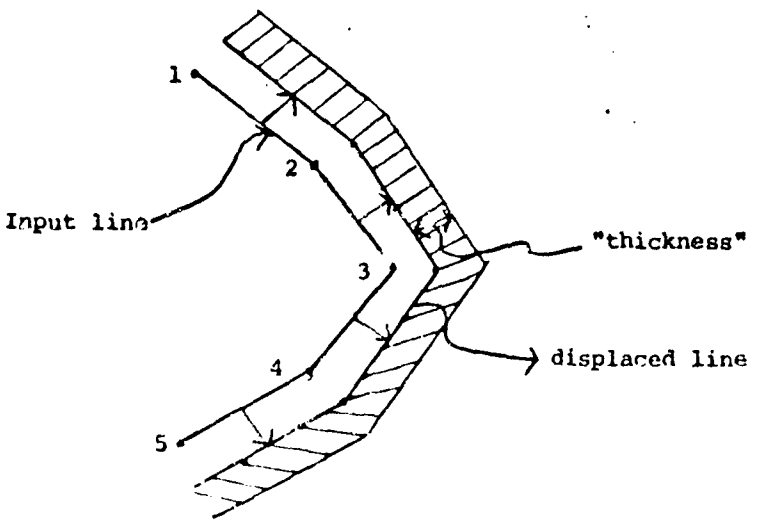
The assignment of trees to cells of the grid requires that cluster densities be established for each tree type. The cluster density is sampled for a grid point which will act as an "anchor" point for location of a cluster. The cluster size is then determined from the cluster mean and a highly peaked distribution function which covers a range from unity to twice the mean. A square array of grid points is then investigated for unoccupied sites and the members of the cluster are assigned to these with probability that decreases with distance from the center. This has the effect of softening the corners. It is possible that some members of the cluster will not be accommodated in the array, but this loss is compensated at the end of the assignment process by distributing individual trees. The dimension of the square array is determined from the expression

LENGTH=SQRT(2*(KL+2))

FIGURE 3.1 - Boundaries and "Thick" Lines



(a) A closed boundary: points 1 through 5 lie along a road bed. Non-zero values of W(1) through W(4) give the appropriate displacements.



(b) A line: points 1 through 5 lie along the same road bed as in (a). By describing these segments in reverse, the displacement is forced in the opposite direction.

where KL is the cluster size. If the mean cluster size is unity this method is by-passed and an individual tree is located at the selected grid site.

A similar treatment is applied to each segment of a "thick" line, with the exception that the members of a cluster occupy successive grid sites along the line, each site selected from a stack which is perpendicular to the line and gives the lines its "thickness". Thus the thickness is used to "scatter" the trees perpendicular to the line.

The fractional density given by the user for each tree type can now be defined: for the bounded region it is the fraction of grid sites that will be occupied by this type; for the line it is the fractional number of stacks which is occupied by trees of this type, and one in a stack. If a line is composed of trees of more than one type it is possible that a stack will have more than one occupant.

The size of a tree is chosen from a flat distribution about the given mean and of half-width given by the user. The polar angle of the trunk is chosen uniformly within a cone whose half-angle is 0.1 radian. Both the azimuth of the trunk and the axial rotation about the trunk are chosen from uniform distributions between 0.0 and 2π. The location of the center of the base of the containing box is selected uniformly within a circle about the grid site with radius equal to half the grid spacing.

Each tree associated with a point set is fully described by the user except for the axial rotation about the trunk. This latter parameter is selected as for trees in the other configurations. However, it was felt that a user might wish to describe a fallen or leaning tree in some specific azimuthal direction and specification of the polar and azimuthal angles is necessary for this. If these entries are left blank the trees will be upright but the axial rotations will give sufficient variation in appearance.

A rough printer plot capability is provided within the program and is applied to each bounded region, each segment of a line and each point set. For the bounded region and for the line segment (always represented horizontally) the technique is to identify grid points with printer positions and for each occupied grid site to print the tree identifier which is an integer between 1 and 6. The same procedure is used for representing the point set except that the grid size is supplied by the user.

A better plotting capability is provided outside the program by the PROGRAM PLOTFR. The card input to the program is a single card giving the rectangular limits of the plot and a grid size. PLOTFR operates on the output tape of FORGEN to provide an integrated view of the specified region.

The identification of a tree on the output tape of FORGEN does not, of course, distinguish between the high resolution model and the low resolution model for that tree. That determination must be made for each scene, i.e., for each ratio of tree-size to distance from the camera. However, the criterion for switching from one model to the other may properly be influenced by the tree situation. For example, a tree in the interior of a dense forest need not be represented as carefully as a single specimen. With this in mind, FORGEN tags such trees by attaching a minus sign to an intrinsically positive quantity, namely, the vertical component of the height of the containing box. The basis of this assignment is that the tree must be a non-boundary tree of a bounded region for which the fractional area occupied by trees is greater than 0.2.

3.3 Description of Routines

In this section, we describe all of the routines of FORGEN. A linkage chart is given in Figure 3.2.

PROGRAM FORGEN

FORGEN governs the generation of the forest library and performs many of the required calculations. In particular, it distinguishes between the generation of a new library and the updating of an old library, and reflects the organization of the calculation according to boundaries, lines and points.

a) Main program

b) Subroutines called: DISPL, INPROC, GRID, ASSIGN, TCAL, BOXIT, POSIT, PLOT1

SUBROUTINE ASSIGN

ASSIGN is called from the main program for boundaries and thick line segments after the grid has been established. It's function is to select a cluster size and a cluster position and to assign the members of the cluster to grid sites. ASSIGN does this for the number of clusters required to provide the calculated cluster density. Because the number of necessary sites are not always available in the cluster, ASSIGN finds individual sites to complete the assignment. Since the assignment process is different for boundaries and line segments, a parameter NOPT (=0 for boundaries, =1 for line segments) is passed to ASSIGN from the main program.

a) Called from: FORGEN

b) Subroutines called: CLSTR

SUBROUTINE BOXIT

This subroutine calculates the box parameters, i.e., the vertex vector and three edge vectors, for an individual tree, assuming an elevation of z=0.0 for the base. It selects an axial rotation angle and accepts from the calling routine a horizontal base position, a polar angle, an azimuth and a magnification of the containing box for that tree type.

a) Called from: FORGEN, TCAL

b) Subroutines called: None

SUBROUTINE CLSTR

CLSTR selects a cluster size and returns this value to ASSIGN. The distribution function is symmetric about the mean with range from unity to one less than twice the mean. The probabilities decrease by a factor of one half with each additional unit departure from the mean.

a) Called from: ASSIGN

b) Subroutines called: None

SUBROUTINE GRID

It is the function of GRID to set up (for bounded regions only) the array which defines the bounded region in terms of grid indices. LIST(1,I) is the lowest index (J1) for horizontal grid lines passing through the vertical grid line of index I, while LIST(2,I) is the highest such index (J2). LIST(3,I) is the cumulative sum of (J2-J1+1) through I-1 and thus constitutes a pointer to an array carrying the grid site assignments. The process is so trivial for thick line segments and for point sets that it is performed in FORGEN.

a) Called from: FORGEN

b) Subroutines called: None

SUBROUTINE INPROC

This subroutine is called from FORGEN to read the data for all the trees of a bounded region or a line, i.e., number of types, type identifier, mean cluster size, fractional density, mean height of containing box, relative range of heights for this region, and the shrink factor. It then calculates the cluster probabilities mean magnifications (ratio of input height to resident box height for that tree type) and the grid size.

a) Called from: FORGEN

b) Subroutines called: None

SUBROUTINE PLOT1

PLOT1 is used to display on the printer the grid site assignments for each region. The position in a printer line corresponds exactly to the x-index I, while the line itself corresponds to the y-index J. The grid spacing is calculated for the boundary or line segment but is an input for the point set. PLOT1 operates from the arrays LIST and IQ. LIST has already been described. IQ(K) contains a zero for an unoccupied site and an identifying integer from 1 to 6 for an occupied site. LIST(3,I) is a value of K such that K+J is the location in IQ corresponding to the (I,J) grid site. Segments of lines are plotted horizontally, but actual end points are printed out.

a) Called from: FORGEN

b) Subroutines called: None

SUBROUTINE POSIT

This routine is called by FORGEN when the code is being operated in the updating mode, i.e., the mode in which an existing library is to have certain elements replaced and other elements added. POSIT must copy acceptable elements from the old tape to the new tape and position the old tape past unacceptable elements.

a) Called from: FORGEN

b) Subroutines called: None.

SUBROUTINE TCAL

TCAL governs the calculation of tree box parameters for boundaries and lines. It selects the final location of the base of the tree within the grid cell, for all trees of ray regions, a magnification within the input limits about the mean magnification, a polar angle and an azimuth. BOXIT is then called to complete the calculation for one tree. TCAL also writes out on tape (TAPE2) the 12-word tree descriptions in records of 200 trees. The calculation is ordered on tree types.

a) Called from: FORGEN

b) Subroutines called: BOXIT

3.4 Input and Logical Unit Assignments

All input data are read from cards on one of two formats:

Integer data: FORMAT(6I10)

Floating Point data: format(6E12.4)

CARD NO.	CONTENTS	COMMENT
1	ISUBST	Mode option parameter equal to 0 or 1. 0: create new library from scratch; 1: update existing library.
	NB, NL, NP	Number of (boundaries, lines, point sets) to be processed for ISUBST=0; number to be replaced for ISUBST=1.

Cards 2,3 and 4 are included only for ISUBST=1.

2	IDB(I),I=1,NB+1	I≤(NB,NL,NP), ordinal number of (boundary, line, point set) to be replaced. I=(NB+1,NL+1,NP+1) number of such regions to be added (may be zero).
3	IDL(I),I=1,NL+1	
4	IDP(I),I=1,NP+1	

Card sequence 5 through 11 is needed for each bounded region and line with the exception of Card 7 which is needed only for lines. Card sequence 10, 11 is repeated for each tree type in each description.

5	NXY	Number of points used to describe this boundary or line.
6	XP, YP	X and Y coordinates of a point;
	W	displacement of line segment between that point and the next. Two points per card.
7	WN	for lines only, "thickness" or allowed straggle of bases.
8	NTYP	Number of tree types for this region.
9	SHRINK	Factor to be applied to tree box width to determine mean spacing of trees in a cluster.
10	ITYP	Tree identifier (11,21,31,41,51,61);
	XLMP	mean cluster size for this type; one type per card.
11	FRCT	Fraction of grid points covered by this tree type;
	SMN	mean height of this tree type; fractional
	SIG	height variation for this tree type. One type per card.

Card sequence 12 through 15 is needed for each point set. Card 15 is repeated for each tree in a point set with order of tree type given by card 13.

CARD NO	CONTENTS	COMMENT
12	NXY NTYP	Number of trees in this set, number of tree types.
13	ITYP NUMR	Tree type identifier; number of trees of this type. Three sets per card.
14	XMIN, XMAX	x-limits;
	YMIN, YMAX	y-limits;
	DS	grid spacing
15	X, Y	x,y coordinates of tree base;
	SMN	height of this tree;
	THET	polar angle of stem (rad);
	PHI	azimuth of stem (rad).

A final note on input cards 5 and 6 for boundaries: the closing is produced by the code on the assumption that the last point is not a repetition of the first point. For both boundaries and lines, the quantities W(I) should be made somewhat greater than half the width of the road or stream bed since it is the base of the tree which is confined to be within the boundary or within the "thickness" of the line. Finally, the points defining a boundary must be sequential and in counter-clockwise order, while those defining a line must be described sequentially in a direction such that the stream or road bed is on the right. The W values are all positive.

Logical units:

TAPE1 : existing tree library for update only (ISUBST=1)
TAPE2 : output tree library.

3.5 The Output Tape

The output tape (logical unit 2) is organized by region type: bounded regions, lines and point sets. Within each type the order is that determined on input.

RECORD NO.	CONTENTS	COMMENTS
1	NBT,NLT,NPT	Numbers of boundaries, lines and point sets

Each region is represented by the following sequence of records.

2	IDUM,NTYP,IDUM2	NTYP - number of tree types
	(ITYP(I),NUMR(I),I=1,NTYP)	Type identifier, number for all types
	XMIN,XMAX,YMIN,YMAX	Rectangular limits of region

Each tree type in a region is represented by the following sequence of records:

3	(TT(I,J),I=1,12),J=1,200)	Up to 200 tree boxes for first tree type in each record.
.	"	
.	"	
L1=2+(NUMR(1)/200+1)	"	

Each tree type starts a new record.

4. THE TERRAIN PROGRAM

4.1 Introduction

The terrain model that was developed under a previous contract[1] was a modification of the camouflage net model[2] that embodied softened contour lines and a "road" capability. It was, however, wedded to inflexible color pattern boundaries, characteristic of the nets, and to the input mode of the net, which is quite inappropriate to a terrain model. The purpose of the effort reported here was to remove these restrictions from the model, to separate the establishment of the terrain library from the ray tracing process, and to provide for easy retrieval of the data for ray tracing. The first two steps required the development of the stand-alone program, PROGRAM INSTAL, and the replacement of a single routine in the ray tracing module. The last step is a matter of positioning the data base tape and reading the processed data into memory.

A further deficiency of the old model was an extremely slow trianglulation method which, in the course of establishing the real terrain of nearby one hundred points, became intolerable. Modification of the subroutine SEGNET was necessary to overcome this problem.

4.2 PROGRAM INSTAL

4.2.1 Design

This is a stand-alone program for reading and processing elevation data, road data, color boundary data and region number assignments. The output is a file which is to constitute the third library on the data base tape. The file consists essentially of the common blocks required by the ray tracing routines.

In the previous model, the terrain installation process took place in two steps: 1) establishment of the flat "net" description in 2-space; 2) construction of the installed "net" in 3-space and calculation of parameters for mapping a (strike) point on the installed net onto a point of the flat net for (struck) region identification. PROGRAM INSTAL replaces the first step entirely and modifies some portions of the second step so that simple elevation data are used in the installation process.

The "flat net" is essentially replaced by supplying a projected region boundary, i.e., a sequence of points (x,y) in a horizontal plane defining a closed boundary on the elevated terrain. Any terrain not enclosed by such a boundary is lumped under a single region assignment. In the ray tracing procedure, the "mapping" problem is replaced by the problem of determining, for given (x,y), the corresponding boundary number.

The treatment of roads has not been changed except that the road input reading and processing is now part of PROGRAM INSTAL.

4.2.2 Description of Routines

Table 4.1 lists the routines of PROGRAM INSTAL and identifies them as new, modified or unchanged. Only the first two categories will be described here.

PROGRAM INSTAL

This is the main program whose function is to call the specific input reading and processing routines and to write the processed data (essentially the common blocks) out onto TAPE8.

a) Main program

b) Subroutines called: INPROS, SEGNET, TRINET, FLIP, NETBOD, ROADIN, INBOUN

SUBROUTINE INPROS

This routine reads elevation data (x,y,z) for LSUP interior points, and locations (x,y) of NUM sea level (z=0.0) points on an enclosing boundary. The output in the HPT array containing, for up to 100 points, the five word group x,y,z,u,v where u=x and v=y. This redundancy, produced by replacing the flat net description, does no harm and allows us to retain the other processing and ray tracing routines intact.

Because elevation values (z) are usually provided in feet, while horizontal locations and distances are given in meters, the code assumes this mixed unit input and divides the elevations by 3.048 ft/meter.

a) Called from: INSTAL

b) Subroutines called: None

SUBROUTINE SEGNET

This routine receives the HPT array from INPROS and performs the triangulation of the terrain. The method is to form all possible segments between pairs of interior points and to order the segments according to increasing length for each point. The shortest segment is accepted as a triangle side if it does not cross any previously accepted segment and is not parallel to a previously determined side of the current triangle. The necessary improvement introduced into this routine is a sort on segment length prior to the point by point search for triangle sides.

a) Called from: INSTAL

b) Subroutines called: None

TABLE 4.1 - The Routines of PROGRAM INSTAL

NAME	COMMENTS
INSTAL	Main program; new
INPROS	Modified
SEGNET	Modified
TRINET	Old
FLIP	Old
NETBOD	Old
ROADIN	Old
INBOUN	New
CROSS	Old
NORM	Old
INIPOS	Old
XYZ2UV	Old

SUBROUTINE INBOUN

This routine reads and processes region boundary data and makes region number assignments according to input. A sequence of NXY points, (x,y) counter-clockwise about a closed boundary (bad point not repeated) is read for each of NB boundaries. A pair of direction cosines is computed and stored for each line segment. The rectangular limits of the boundary are also stored to speed up the search process which takes place in ray-tracing. A list of NB+1 color region numbers is read and stored, the last corresponding to all uncovered terrain.

a) Called from: INSTAL

b) Subroutines called: None

4.2.3 Input, Output and Logical Unit Assignments

Card (1) LSUP, NUM FORMAT(2I5)

LSUP - Number of interior points for elevation data

NUM - Number of boundary points with z=0.0. Read by INPROS.

Card (2) x,y,z FORMAT(3F10.0)

Elevation data for LSUP points, 1 point per card. Read by INPROS. Scale difference between (x,y) and z assumed: (x,y) in meters, z in feet.

Card (3) x,y FORMAT(2F10.0)

Locations of NUM boundary points, 1 point per card. Read by INPROS.

Cards (4) through (6) Read by ROADIN - see (1) p. 43.

Card (7) NB FORMAT(I5)

Number of region boundaries to be read. Read by INBOUN.

Card sequence (8) and (9), read by INBOUN, repeated for NB boundaries.

Card (8) NXY FORMAT(I5)

Number of points for this boundary

Card (9) XP,YP FORMAT(2F10.0)

Coordinates of NXY points, 1 point per card.

Card (10) ICOLOR(I),I=1,NB+1 FORMAT(14I5)

Ordered color numbers for NB bounded regions and "rest of world". 14 per card. Read by INBOUN.

Logical unit assignment: TAPE8 = output tape.

The output tape has nine records, each of which contains the data for a labeled common block or data for unlabeled common as used in the ray tracing elements of the model. The records are as follows:

RECORD NO.	CONTENTS	COMMON
1	LSUP,IPEG,LINT,HPT(500)	LIMAGE
2	LIJ(600)	HSEGNE
3	NT,LETRIP(600)	HTRINE
4	NV,HBD(4000)	HTRKNE, unlabeled common
5	JIT,CJIT,LRN	COMRN
6	NUMROD,NHIT,NCROAD	ROADC1
7	IPROAD(25),IPNORM(25),NSEG2(25) WIDTH(25),WIDTHS(25),NCR(25), LRUT(25),FRAC(25),CRAC(25)	ROADC2
8	XP(1000),YP(1000),WX(1000), WY(1000),NP(100),NB	POINT
9	XLIST(400),ICOLOR(100)	ORGAN
10	EOF	

4.3 Terrain Ray-Tracing

The ray-tracing module of the new terrain model differs from that of the old model, described in a and b, only in the replacement of the routine FIND by a pair of routines, TFIND and LOCAL (called by TFIND).

SUBROUTINE TFIND .

This routine is called from RAYTRK after a ray has struck the terrain at a point (x,y,z). Using the coordinate x and y, FIND examines the rectangular limits of each of the bounded regions to determine whether that region may contain the point. If a region is a candidate, FIND calls LOCAL to perform the search with the real boundary. If a bounded region IR contains the point, the assigned region number, ICOLOR(IR) is extracted and returned to RAYTRK. Otherwise it is assumed that the strike point is in the "rest of the world" and ICOLOR(NB+1) is returned.

a) Called from: RAYTRK

b) Subroutines called: LOCAL

SUBROUTINE LOCAL(IR,IFL)

This routine uses the ordered sequence of points, (X,P(I),YP(I),I=1,NXY), that define the boundary of the region IR and the associated direction cosines, (WX(I),WY(I)), to determine whether the point x,y is within the region. The number of boundary points NXY is extracted from the array of pointers NP. If the point is within the region the signal IFL is set to unity; otherwise a zero is returned to FIND.

a) Called from: FIND

b) Subroutines called: None.

4.4 Data Retrieval for Ray Tracing

The data retrieval for the terrain model is under the control of the executive of the overlay system and is discussed in Section 5. Since the processed terrain data file is the third data library on the IDB, retrieval consists in positioning the IDB (SUBROUTINE TERRET) and reading the nine records into memory (SUBROUTINE INTER).

5. THE IMAGE GENERATING SYSTEM

5.1 Introduction

The components of the image generating system have been listed in Section 1.6, but the list will be repeated here for ease of reference:

a) For each of the four geometry models, the ray-tracing module which produces an unshadowed image of the subject geometry.

b) For each of the four geometry models, the ray-tracing module which shadows, with the subject geometry, a previously generated block format image on the user designated tape unit IMFI.

c) A scene setting module corresponding to the old CAMERA module and performing similar input reading and processing functions for shadow runs as well as for image generating runs.

d) A data retrieval module whose components and functions have been described in Section 2.

e) A module (PROGRAM CONV) for converting a block format image to a string format image.

f) A module (PROGRAM MERGE) for merging two block format images.

g) An executive whose function is to read option cards and, in response to these option cards, to invoke the sequence of selected runs, while providing through the data retrieval module, the necessary input for each run.

Of the thirteen components named above, only the data retrieval module (a) and the executive (g) are completely new. The vegetation ray tracing of (a) and (b) are complete revisions of the old model as described in Section 1.2. The terrain ray tracing of (a) and (b) contains relatively minor revisions as described in Sections 1.5 and 4.3. For all the modules of (a) and (b), input and output have been altered to read input data from the data base and to output the block format image on a user designated logical unit IBFO.

A basic change in the modules of (a) allows block-by-block merging of the image being generated with a previously generated block format image located on the user designated logical unit IBFI, which, if set to zero, will inhibit the merge. The camera-to-object distances for the points of a block of the established image are used to initialize the maximum search distances for the corresponding rays of the current image. Thus, there is a dual advantage in this procedure: increased efficiency of ray tracing as well as reduced I/O in the generation of a composite image.

All of the modules involved in the reading and writing of block format images have been altered to conform with a change in format introduced to reduce I/O time. The change in the format of the block format image file involves the second of the two records which are written for each block of the image[1]. Formerly this record contained, for each "non-sky" point, four words: an identifier for location of the point in the block, and the product of the camera-to-object distance with each of the components of the object normal at the point struck. These last three words are now packed into a single word, thus substantially reducing the length of the file for a complex image. Changes were thus required in the old PICTUR program of the modules of (a) and in the old SHADOW program of the modules of (b). In addition PROGRAM CONV and PROGRAM MERGE were modified.

The scene setting components of (c) are derived in part from the old CAMERA module and in part from the old SHADOW program. One function is to read, process and store those input data which are common to the elements of a scene: camera location and direction, focal length, image dimensions, grid

characteristics or, for shadow runs, source type, source direction and so on. A second function is to determine the ground plane limits for retrieving the trees which appear in the scene. On the other hand the data which are peculiar to the geometry model are read by the managing routines of that model: picture number, location and scale factor for targets or clets, raster limits for image generation or shadowing.

The remainder of this section will be devoted to a discussion of the executive (Section 5.2), a description of the overlay structure (Section 5.3), a description of new subroutines of the system (Section 5.4), and finally, a user's guide with input description and logical unit assignments (Section 5.5).

5.2 The Executive

The executive can best be described in terms of the flow chart of Figure 5.1 and a glossary of program variables which will be given here:

IDB — the logical unit (set to 1) on which the data base resides.

IPD — the logical unit (user input) which is to be used to transfer selected data between the IDB and the data organizing programs, (PREPAR) of the vegetation modules. This variable is also used as a flag for the scene setting elements: zero for image runs and unity for shadow runs (ISAVE=1).

IA — Hollerith data supplied by the user on the first option card. Sets the value of ISAVE according to the following table:

IA	ISAVE	Path
ICAM	1	set scene
IGEO	2	generate an image
ISHA	3	shadow an image
ICON	4	convert an image
MERGE	5	merge two images
IEND	6	rewind tapes and STOP

IB — Hollerith information, supplied by the user on the first option card. Sets the value of JSAVE which selects among the four geometry models for a geometry run (ISAVE=2 or 3) and may serve to warn the scene setting program (ISAVE=1) that a vegetation run is contemplated.

IB	JSAVE	Model
ICG	1	CG target
INET	2	camouflage net
IVEG	3	vegetation
ITER	4	terrain

IBFI — Generic name for an input tape, integer value supplied by the user on the second option card. For an image generating run, IBFI may be zero, or it may point to a block format image with which the current image is to be merged as it is generated. For a shadow run IBFI desingates the unit of the image to be shadowed. For a CONVERT run, IBFI designates the unit of the image to be converted. Read as IF(1).

IBFO — Generic name for the output tape unit, user supplied, for any run; must be non-zero for ISAVE=2,3,4, or 5. Read as IF(2), except for ISAVE=5, where it is read as IF(3).

ISV1 — Generic name for a "save" file. If ISV1 ≠0, the output on IBFO is copied to ISV1, file number NFILE and a printout identifie the image, the unit and the file. Read as IF(6), if ISV1 is different from previous value, NFILE is set to zero and advanced by unity just prior to copy. For IA=IEND(ISAVE=6), IF(I) gives the sequence of save files for rewinding.

IS1,IS2 — Generic names for units of two block format images to be merged (ISAVE=5). Read as IF(1), IF(2). IS1 may also be used to identify a target or net by sequence number (ISAVE=2,3; JSAVT=1,2). Then IS1 is read as IF(4).

JFIRST,JLAST — The first and last indices of tree models to be processed for a vegetation run, image or shadow. Read as IF(4) and IF(5). Default values (if either is zero) are 1 and 12 respectively. The corresponding tree identifiers are given in Section 2.1.

FIGURE 5.1. FLOW CHART OF THE SYSTEM EXECUTIVE

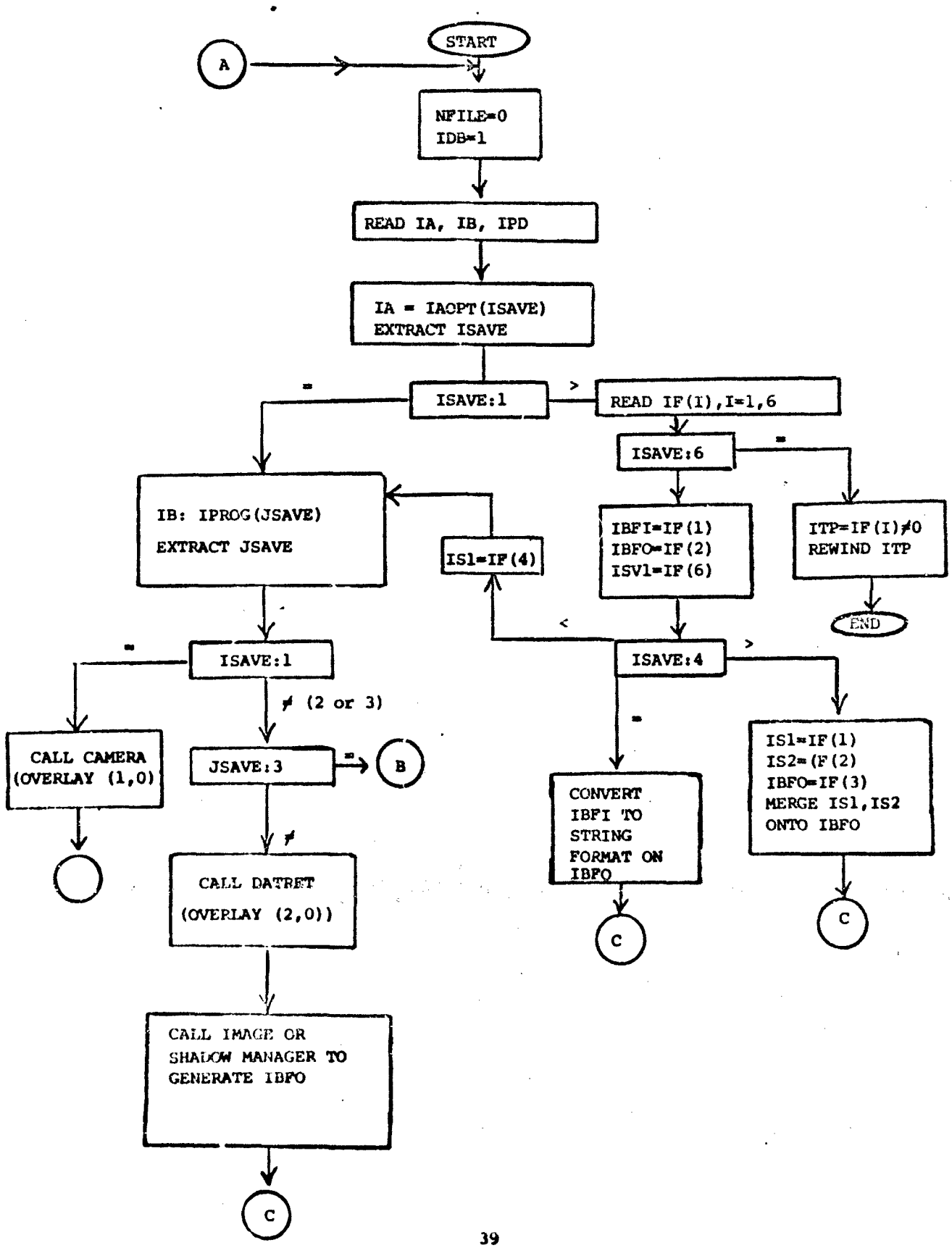
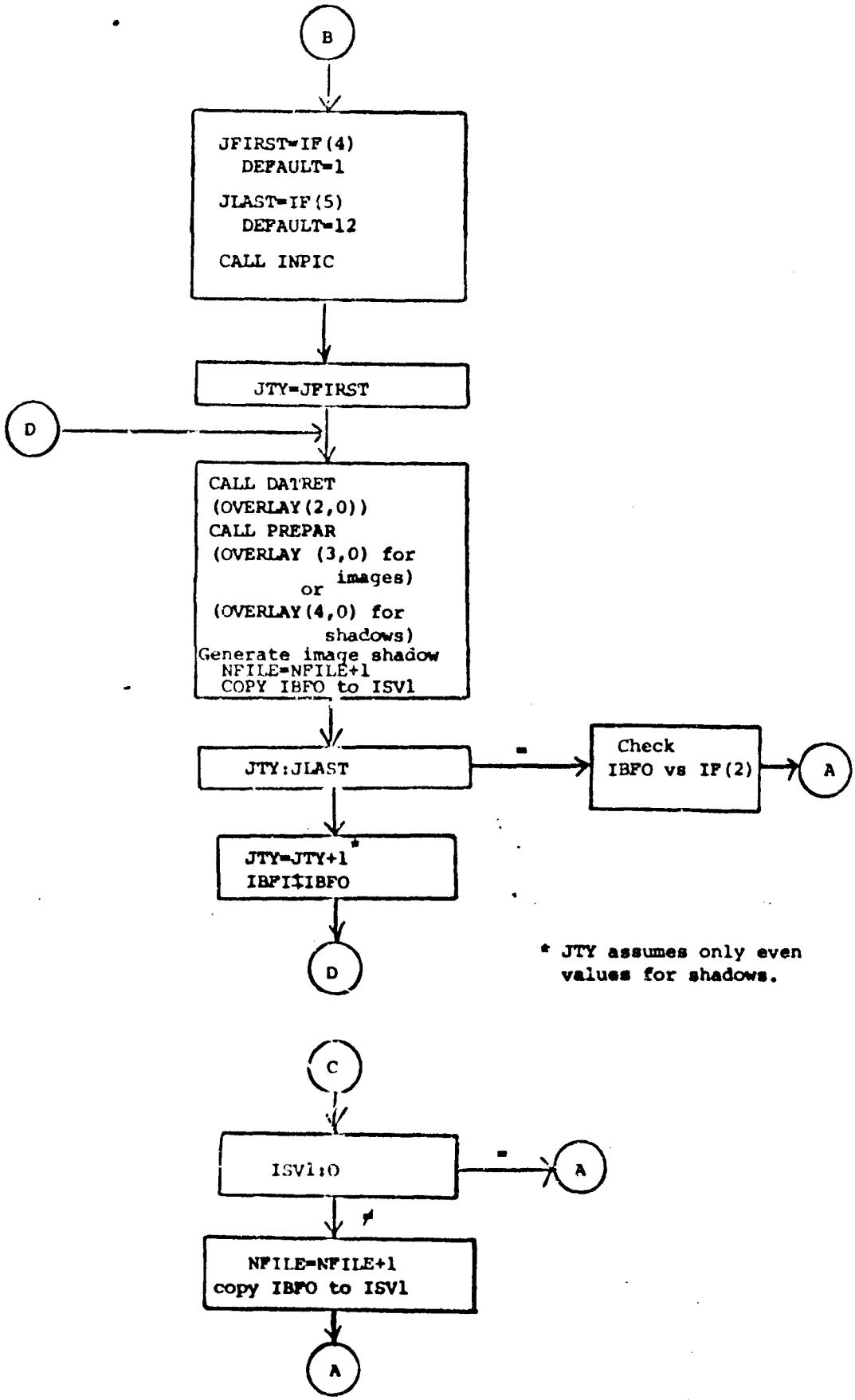


FIGURE 5.1. Continued



Thus, on the first option card, the user selects a given type of run (ISAVE) and, for geometry runs or a scene setting run, a specific geometry (JSAVE). He must also select a value of IPD on this card for two cases:

a) A scene-setting run (ISAVE) for images requires IPD=0; for shadows IPD=1.

b) A vegetation image or shadow run must have a non-zero value of IPD.

On the second option card, required for ISAVE>1, the user selects logical units for input and output and (possibly) for a permanent storage file. For a geometry run for CG targets or nets, he also specifies a sequence number for the particular target or net. For a geometry run for vegetation he may specify a first and last index for tree models to be processed.

Vegetation runs, image or shadow, are handled differently from the other geometries. Because of the different tree models which may occur in a given scene the vegetation run is performed in a loop on tree model sequence number (see Section 2.1 for correspondence of sequence numbers, tree type numbers and model identification numbers). The user may leave control to the program by leaving IF(4) and IF(5) blank. This will invoke a complete set of vegetation runs, i.e., all sequence numbers found for this scene will be run for images and all even sequence numbers (low resolution model) will be used in shadowing. Or, the user may control the run in one of two ways: 1) he may, as mentioned above, select a first and last sequence number for a single exhaustive run, or 2) he may opt for several runs with selected models. Clearly the decision will be based on the user's knowledge of the forest library and the scene limits.

The executive calls eleven primary overlays whose main programs are given in the next sub-section. The subroutines called by the executive and residing with it in the main overlay are INPIC (for reading geometry dependent camera data) and COPYIT (for saving intermediate output on tape or disks).

5.3 Description of the Overlay Structure

The overlay structure for the system is depicted in the block diagram of Figure 5.2. Each overlay is represented by a labeled box which contains a list of subroutines headed by the main program of the overlay. The input and output units are shown on the alternate block diagram of Figure 5.3.

A few words should be said about the sub-program names to aid in identifying similar programs in different overlays. IBM overlay procedure will not maintain identical names in different overlays. The action would be to move the first such module to the next lower level and to eliminate succeeding routines of that name. Hence the method adopted here was to rename in a manner suggestive of the overlay function while retaining enough of the generic name for easy identification. Thus the modules PICTUR, NTPICT, HPICTU, LPICTU are almost identical but serve to generate, respectively, combinatorial geometry images, net or terrain images, high resolution vegetation images and low resolution vegation images.

The main overlay (0,0) of the system contains the executive, PROGRAM TRAFFIC, its subroutines INPIC and COPYIT, and several other subroutines which are used in several branches of the system. Card input is read by TRAFFIC and INPIC.

Several branches of the program terminate with the primary overlay:

(1,0) the scene-setting overlay, main program CAMERA. Cards are read by CAMERA.

(2,0) the data retrieval overlay, main program DATRET. The data base is positioned for later reading or is accessed for selection of data to be output on the IPD.

(3,0) the forest organizing overlay for images, main program PREPAR. The IPD is read by PREPAR and the forest data ordered by image block in records of 200 trees, output on TAPE3. The output TAPE10 contains information to order trees by distance from the camera. Scratch tapes 11, 12 and 13 are used.

FIGURE 5.2. BLOCK DIAGRAM OF THE OVERLAY STRUCTURE

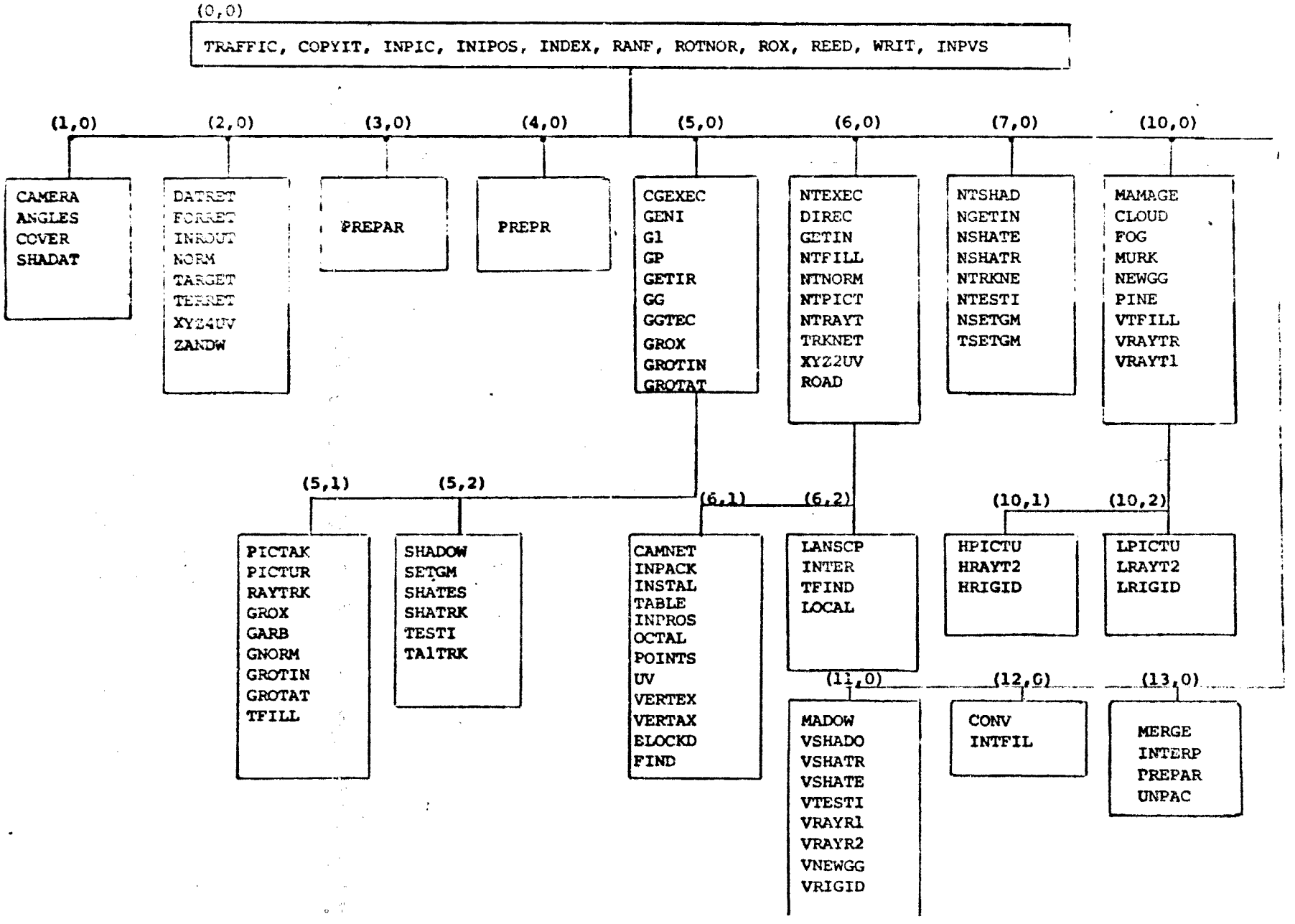


FIGURE 5.3. BLOCK DIAGRAM OF OVERLAY STRUCTURE SHOWING INPUT AND OUTPUT UNITS

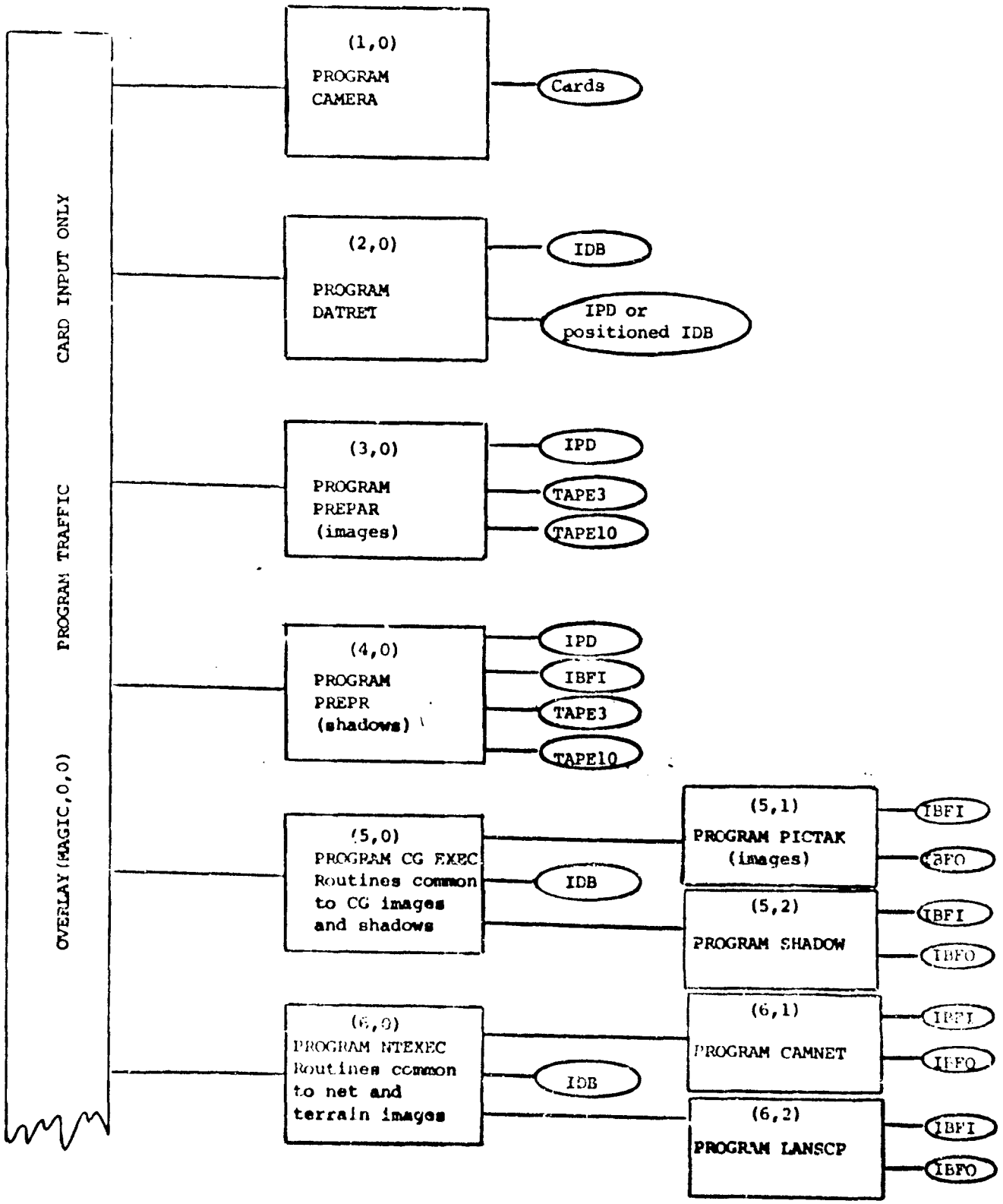
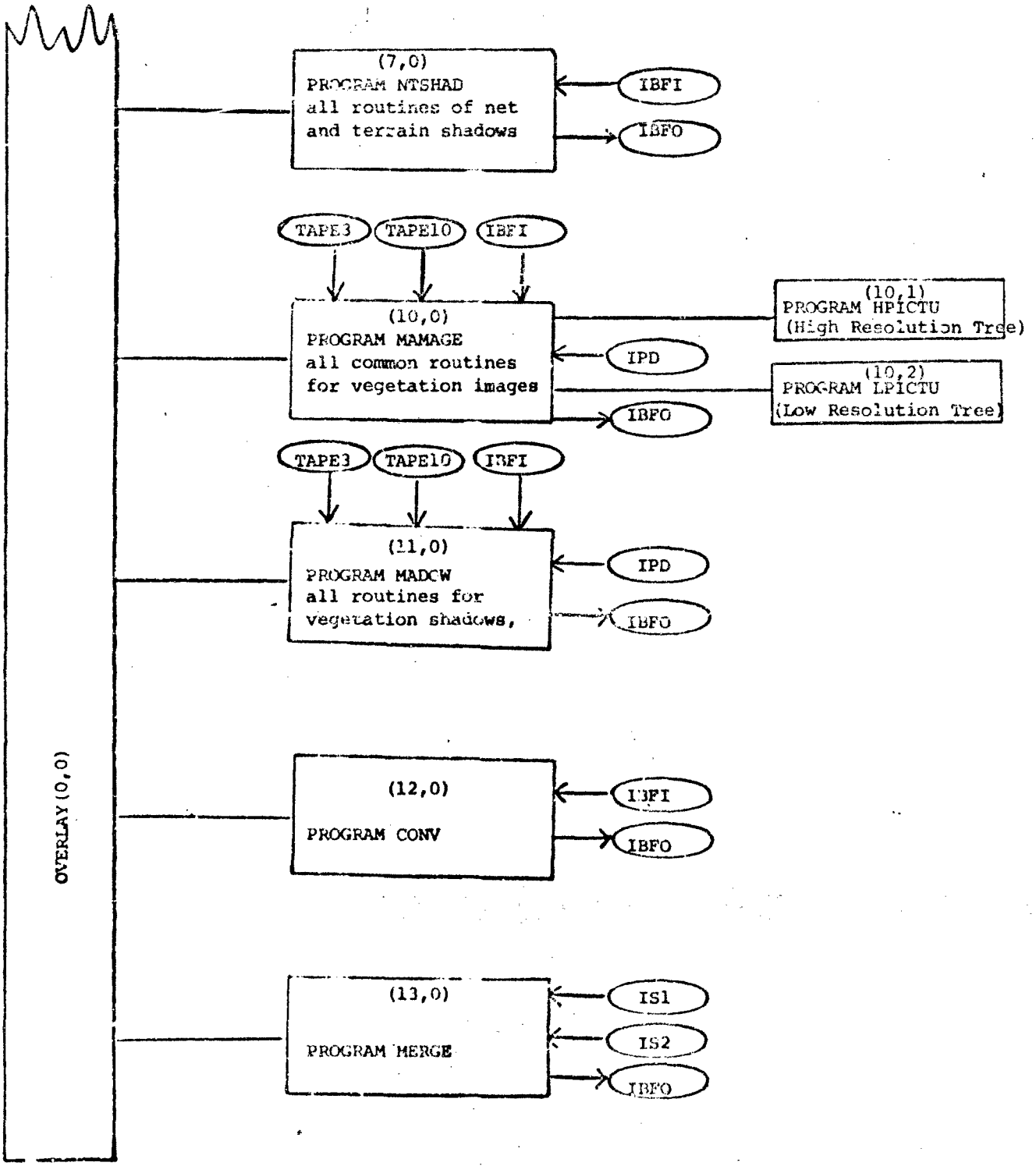


FIGURE 5.3 Continued



(4,0) the forest organizing overlay for shadows, main program PREPR. The IPD, containing forest data and local terrain normals, and the IBFI (to be shadowed) are read by PREPR to determine whether the tree shadow volume intersects the object volume for a given block. Output tapes 3 and 10 are similar to those from PREPAR, except that the distance array on TAPE10 is replaced by distance limits of the shadow volume along the ray. Scratch tapes 11, 12 and 13 are used.

(11,0) the primary overlay for vegetation shadows, main program MADOW. MADOW reads tree model data from the IPD, and calls the routine VSHAD (equivalent to the old SHADOW program) to generate shadows for the trees represented on tapes 3 and 10. Only the low resolution model ray tracing is implemented for shadows.

(12,0) the conversion overlay, PROGRAM CONV, reads IBFI and writes on IBFO the string format version of the input block format image.

(13,0) the merge overlay, PROGRAM MERGE, reads and merges two block format images on IS1 and IS2 and writes the result on IBFO.

The other branches of the program terminate with one of two alternative secondary overlays. In general the primary overlay contains a manager, for discriminating between the two secondaries for satisfying a particular option, and most of the routines that are common to the two options served. Each secondary overlay contains the routines peculiar to its mission and the main program for that mission. The method of branching was chosen to maximize the

number of common subroutines in the primary overlay and hence to minimize repetition of programs. A complete branch is either an image run or a shadow run for one of the geometry models. Each such branch reads some card data (transaction vector and scale for targets and nets, picture number, raster limits and geometry data from the data base and may use an IBFI as an input tape. Output is a block format image on IBFO.

(5,0) the primary overlay for Combinatorial Geometry images and shadows, main program CGEXEC. CGEXEC is the manager which discriminates between images (5,1) and shadows (5,2) on the basis of the value of ISAVE (2 for images, 3 for shadows).

(5,1) the secondary overlay for CG images, main program PICTAK.

(5,2) the secondary overlay for CG shadows, main program SHADOW.

(6,0) the primary overlay for camouflage net and terrain images, main program NTEXEC. NTEXEC is the manager which determines whether to call for a net image (6,1) (JSAVE=2) or for terrain image (6,2) (JSAVE=4).

(6,1) the secondary overlay for net images, main program CAMNET.

(6,2) the secondary overlay for a terrain image, main program LANSCP.

(7,0) the primary overlay for camouflage net and terrain shadows, main program NTSHAD. NTSHAD discriminates between net shadows (10,1), (JSAVE=2), and terrain shadows (10,2), (JSAVE=4).

(7,1) the secondary overlay for net shadows, main program NSHAD (based on SHADOW).

(7,2) the secondary overlay for terrain shadows, main program TSHAD.

(10,0) the primary overlay for vegetation images, main program MAMAGE. MAMAGE calls the appropriate secondary overlay for a high resolution model (10,1) or for a low resolution model (10,2).

(10,1) the secondary overlay for high resolution tree images, main program VPICT1 (old PICTUR).

(10,2) the secondary overlay for low resolution tree images, main program VPICT2.

5.4 Description of New Routines

This material will be organized more or less by overlay and program branch. We start with the main overlay, referring the reader to Section 5.2 for the description of the executive.

Other new routines of the main overlay are:

SUBROUTINE INPIC

This routine reads, processes and stores geometry-dependent camera data for image generation: picture number, location and scale of a target, raster limits for image scanning.

a) Called from: TRAFFIC (for vegetation), PICTAK, CAMNET, LANSCP.

b) Subroutines called: None

SUBROUTINE INPVS

This routine reads, processes and stores geometry-dependent data for vegetation shadow generation: picture number of shadowing geometry, location and scale of shadowing geometry, raster limits for shadowing.

a) Called from: TRAFFIC

b) Subroutines called: None

SUBROUTINE COPYIT (TP1,TP2,ISIG)

For ISIG=0, COPYIT copies a block format image on TP1 to TP2. For ISIG=1, COPYIT copies a string format image from TP1 to TP2.

a) Called from: TRAFFIC

b) Subroutines called: None

In OVERLAY (1,0) the following subroutines are new or revised:

PROGRAM CAMERA

This is a revised version of the old program, and is flow charted in Figure 5.4. A "CAMERA run" is an option which must be invoked before a sequence of image runs and before a sequence of shadow runs. If a vegetation run is contemplated in either sequence, the program parameter IB must be set to IYEG (see Section 5.2). Then JSAVE will be set to 3 by the executive. JTY (read as IPD on the first option card) is zero for image runs and unity for shadow runs. For JSAVE=3 and any JTY, camera data (position, direction, focal depth, focal plane dimensions, grid specifications) must be available and is read and processed by CAMERA before the ground plane limits for vegetation are determined by SUBROUTINE COVER. For JSAVE≠3 and JTY=0 the camera data are read and processed, but nothing further is done For JSAVE≠3 and JTY=1, no camera data are needed, but sun data (source type parameter and direction of a monodirectional source or position of a point source) are read.

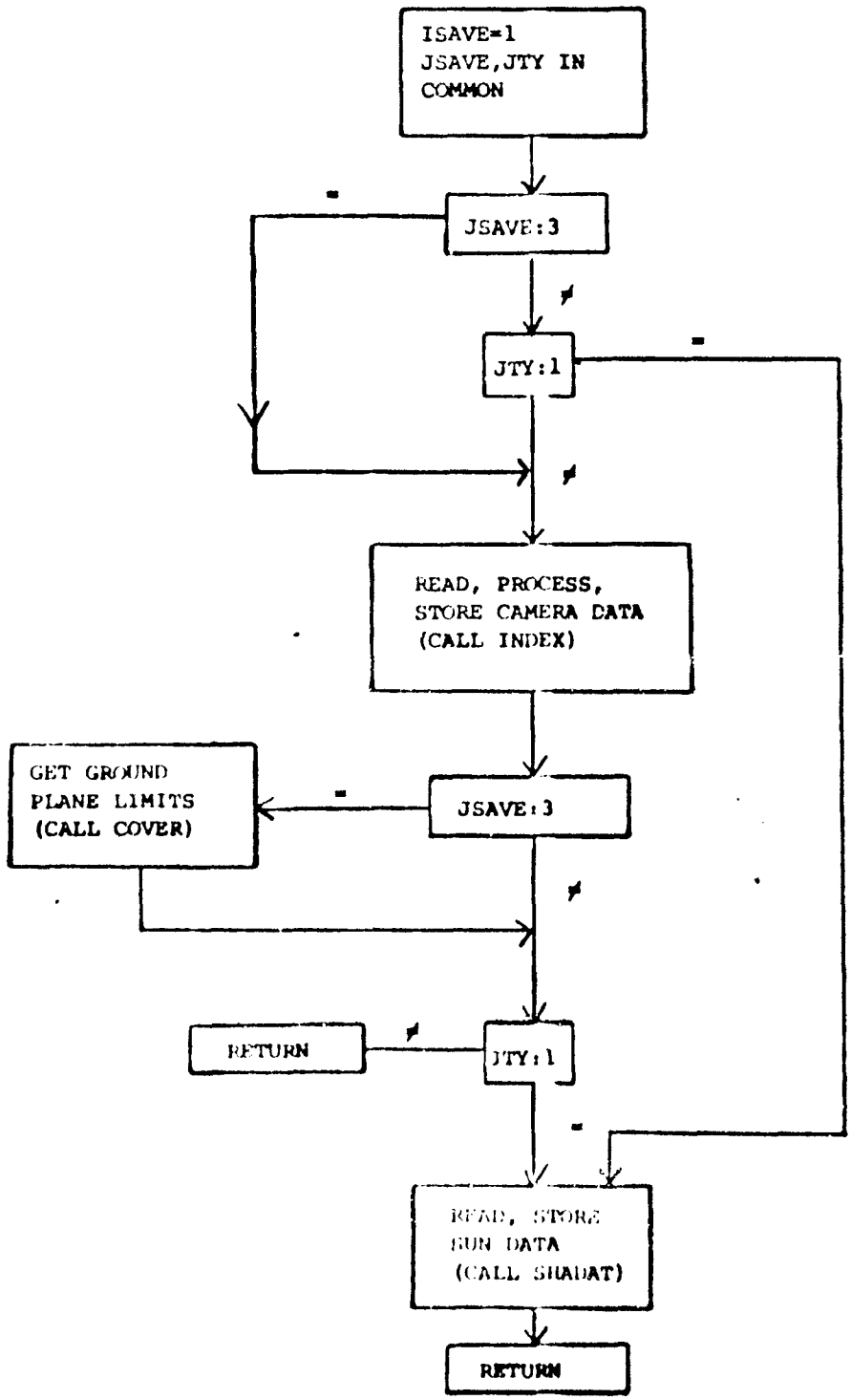
a) Called from: executive, as SUBROUTINE CAMERA for IBM and as OVERLAY(1,0) for CDC.

b) Subroutines called: ANGLES, COVER, SHADAT

SUBROUTINE COVER

This routine is called by CAMERA, if IB=IVEG, to find four corners of a ground plane (sea level) trapezoid which is to be the search area for trees

FIGURE 5.4. CAMERA: The Scene Setting Overlay



for the current scene. COVER calls INDEX to find the direction cosines of limiting rays and thus establishes corners as intersections of these rays with the ground. A corner will be at infinity if its ray does not strike the ground. If all four corners are on one side of the vertical plane through the camera facing the scene, the near corners are moved to that plane along the sides of the trapezoid. Some parameters are stored for the search process.

a) Called from: CAMERA

b) Subroutines called: INDEX

SUBROUTINE SHADAT

Called from CAMERA if JTY=1, SHADAT reads sun data.

a) Called from: CAMERA

b) Subroutines called: None

The routines of OVERLAY(2,0) have been described in Section 2.2.

The routines of OVERLAY(3,0) and OVERLAY(4,0) have been described in Section 2.3.

In OVERLAY(5,0) only the manager is new. The input reading routines GENI and GROX have been altered to read card images from the data base.

PROGRAM CGEXEC

This is the main program of OVERLAY(5,0), which contains the routines common to CG image generation and shadowing. CGEXEC receives the option flag ISAVE (=2 for images, =3 for shadows) through common and calls the appropriate secondary overlay.

a) Called from: executive (as OVERLAY(5,0)

b) OVERLAYS called: (5,1) for images and (5,2) for shadows

OVERLAY(5,1) contains the routines specific to CG image generation, main program PICTAK. PICTAK, the old CG image manager, has been altered to call INPIC and to read card images from the data base. The SUBROUTINE PICTUR has been al-

tered to allow merging of the image as it is generated with the block format image on a user designated IBFI.

OVERLAY(5,2) contains the routines specific to CG shadowing, main program SHADOW.

PROGRAM SHADOW

This is a revision of old PROGRAM SHADOW[1] to remove reading of universal sun and camera data (now done by SHADAT and CAMERA) and to shadow the image on IBFI and output, the shadowed image on IBFO. This SHADOW reads geometry dependent card input: picture number, origin and scale of geometry, and raster limits for shadowing.

a) Called from: OVERLAY(5,0)

b) Subroutines called: SETGM, SHATES, REED, WRIT, TESTI

In OVERLAY(6,0), which contains routines common to camouflage net and terrain image generation, only the manager is new. The names of routines used in other parts of the system have been changed to satisfy the IBM requirement of unique naming. The old name is preceded by "NT" and the whole truncated to six letters. (PICTUR replaced by NTPICT, etc.).

PROGRAM NTEXEC

This is the main program of OVERLAY(6,0). It receives the flag, JSAVE, through common, and decides between camouflage nets (OVERLAY(6,1), JSAVE=2) and terrain (OVERLAY(6,2), JSAVE=4).

a) Called from: executive

b) Overlays called: (6,1) for camouflage nets, (6,2) for terrain.

OVERLAY(6,1) contains the routines specific to camouflage net image generation, main program CAMNET. The old CAMNET has been altered to call INPIC and to accept the input for restart option number 5 from the data base.

SUBROUTINE INSTAL

This routine reads the previously processed net data from the positioned data base (instead of the old restart tape) and then rewinds the data base file.

a) Called from: CAMNET

b) Subroutines called: None

OVERLAY(6,2) contains the routines specific to terrain image generation, main program LANSCP. All routines are new.

PROGRAM LANSCP

This is the manager program for terrain images which calls the various subroutines in sequence.

a) Called from: NTEXEC (as OVERLAY(6,2))

b) Subroutines called: INPIC, INTER, NTPICT.

SUBROUTINE INTER

INTER reads the nine records of processed terrain data from the data base and rewinds the data base.

a) Called from: LANSCP

b) Subroutines called: None

SUBROUTINE TFIND

This is the terrain version of FIND which is described in Section 4.3.

SUBROUTINE LOCAL

This routine has been described in Section 4.3.

OVERLAY(7,0) contains all of the routines for camouflage net and terrain shadows. NTSHAD is the main program and is identical to the PROGRAM SHADOW described for OVERLAY(5,2). Only the data reading routine, SETGM(1) has been changed.

SUBROUTINE NSETGM

This is the camouflage net version of SETGM. It reads processed data from the positioned data base file and computes and stores some parameters for ray tracing.

a) Called from: NTSHAD

b) Subroutines called: None

SUBROUTINE TSETGM

This routine differs from NSETGM only in the nature of the reads from the positioned data base. For shadowing by terrain, only the first four records of processed data are needed.

a) Called from: NTSHAD

b) Subroutines called: None

OVERLAY(10,0) contains all of the routines common to high resolution and low resolution vegetation image generation, with the exception of the old SUBROUTINE PICTUR (see discussion of secondary overlays). The main program is MAMAGE. Many of the ray-tracing routines are new but have not yet been documented.

PROGRAM MAMAGE

This program reads the tree model data into memory from the IPD, identifies the model as high or low resolution and calls the proper overlay for ray tracing.

a) Called from: executive, as OVERLAY(10,0)

b) Overlays called: (10,1), (10,2)

OVERLAY(10,1) contains, as main program, the revised and renamed subroutine PICTUR, called here PROGRAM HPICT. The subroutines HRAYT2 and HRIGID are respectively, the subroutines RAYTR2 and RIGID for high resolution ray tracing.

OVERLAY(10,2) contains, as main program, the revised and renamed subroutine PICTUR, called here PROGRAM LPICT. The subroutines LRAYT2 and LRIGID are, respectively, the subroutines RAYTR2 and RIGID for low resolution ray tracing.

OVERLAY(11,0) contains all of the routines for shadowing by a low resolution model of a tree. The main program is MADOW.

PROGRAM MADOW

This program reads the tree model data into memory from the IPD and calls the routine, VSHADO, which governs the scanning of the image being shadowed. If it proves desirable, in the future, to introduce the high resolution model for shadowing, this is the appropriate routine to use as a manager to distinguish between high and low resolution ray tracing.

a) Called from: executive, as OVERLAY(11,0)

b) Subroutines called: VSHADO

OVERLAY(12,0) is the old PROGRAM MERGE, described in reference 1, and revised to read from the user designated files IBFI and IS1 and write the output on a user designated file IBFO.

OVERLAY(13,0) is the old PROGRAM CONV, described in reference 1, and revised to read and convert the user designated file IBFI and output the result on the user designated file IBFO.

5.5 Input and Logical Unit Assignments for the Image Generating System

The card input to the image generating system is of two kinds: 1) option cards, two for each option that is invoked, except for the camera option, which requires only one, and 2) card input for scene setting, specific to each option. Option cards are read by the executive and are used to select the program and to designate logical units for input, output and permanent storage. The other card input is read by some subroutine called by the selected program.

The following logical units are permanently assigned to input or to a program and should not be used elsewhere except with great caution and knowledge of the system:

TAPE1	assigned to the data base; should never be used elsewhere.
TAPE3, TAPE10, TAPE11, TAPE12, TAPE13	used by the forest data organizing overlays, (3,0) and (4,0), these tapes should not be used elsewhere during a vegetation run.

Any other units named in the program card of PROGRAM TRAFFIC are at the disposal of the user, as described in Section 5.2 and in the input description which follows:

OPTION CARD NO.	CONTENTS	FORMAT
1	IA, IB, IPD (see Section 5.2)	(A4, 6X, A4, 6X, I10)
2	IF(I), I=1,6 (see Section 5.2 and below)	(6I10)

IA=ICAM

There are three possibilities:

1) IPD=0, any I? - camera cards are required.

2) IPD=1, IB=IVEG - camera cards and sun cards are required.

3) IPD=1, IB≠IVEG - sun cards are required.

CAMERA CARD NO.	CONTENTS	FORMAT
1	CP, CD, IOPT, NO	(6F10,5X,2I5)
	CP(3) = camera position CD(3) = camera direction IOPT=0) aiming point (IOPT=1)	
	NO = initial ray spacing in grid points (default 8)	
2	FD, LEFP, FPOFF	(5F10.0)
	FD = focal length LEFP(2) = vertical and horizontal dimensions of focal plane	
3	MAX	(2I5)
	MAX(2) = maximum number of raster points vertically and horizontally	

SUN CARD NO.

1 IFLA, DS, WSS (5X,I5,4F10.0)

IFLA = source type flag,
= 0, plane source
= 1, point source

DS = clearance distance from image (for IFLA=0)

WSS(3) = direction from image to source (IFLA=0) or location of point source, IFLA=1.

IA=IGEO

a) Any geometry (any IB) will require the following:

IF(1) =	0	no input image for merging
=	IBFI	logical unit of input image
IF(2) =	IBFO	logical unit of output image
IF(6) =	0	don't save intermediate output
=	ISV1	copy IBFO to ISV1

b) IB = ICG or INET

IF(4) =	IS1	ordinal number of target or net in target or net file.

c) IB = IVEG

IF(3) =	IBFI	if IBFI was 0 for first tree model image, then IF(3) is the output file for the cumulative image with the second tree model. IBFI and IBFO will alternate under program control.
IF(4),IF(5)		JFIRST, JLAST, first and last tree models to be run, see Section 5.2.
IPD		must be specified.

d) All image generating runs require the following cards:

IMAGE CARD NO.	CONTENTS	FORMAT
1	IPIX = picture number	(I10)
2	CEN,SCF	(4F10.0)
	CEN(3) = location of geometry origin in the real world	
	SCF = scale factor	
3	blank card	
4	ISTRT, IEND, JSTART, JEND	(4I5)
	vertical and horizontal raster limits for this image.	

IA=ISHA

a) All shadow runs (any IB) require (see IA=IGEO, above)

IF(1) = IBFI

IF(2) = IBFO

IF(6) = 0 or ISV1

b) IB = ICG or INET

IF(4) = IS1

c) IB = IVEG

IF(4), IF(5) JFIRST, JLAST
IPD must be specified.

d) All shadow runs (any IB) require the same cards as image runs, except for the blank card. It should be noted, however, that the raster limits here should cover the expected shadow area on the image being shadowed and will, in general, be different from the limits for the image generating run. IPIX refers to the shadowing geometry and is not used except for identification on printout.

IA=ICONV

a) IB and IFD are irrelevant to conversion runs.

b) Conversion runs require

IF(1) = IBFI

IF(2) = IBFO

c) The following additional cards are required:

CONVERSION CARD NO.	CONTENTS	FORMAT
1	NPICT, ICV, ALF, WS(3)	(2I10, 4F10.0)

NPICT = highest value of IPIX

ICV = flag: <0 printer display only

= 0 display + tape output

>0 tape output

ALF = fraction of light from sun; remainder, sky light

WS(3) = vector pointing toward sun

Card 1 is followed by NPICT groups of cards:

CARD NUMBER	CONTENTS	FORMAT
1	NRS	(I10)
	NRS = number of regions for this picture number (=0, for absent picture)	
2*	NCOLOR(I), I=1, NRS	(10I5)
	user supplied color number versus region for this picture number.	

*Card 2 supplied only for non-zero NRS.

IA=MERG

a) IB and IPD are irrelevant to merging

b) Merging requires:

IF(1) = IS1	logical unit of first input image
IF(2) = IS2	logical unit of second input image
IF(3) = IBFO	logical unit of output image

c) No additional cards.

IA=IEND

a) (IF(I),I=1,IMAX) are logical units being used to save output. The named files are rewound.

6. DEMONSTRATION OF THE SYSTEM

All of the major options of the system have been exercised successfully in test problems devised by MAGI. In addition, a large scale demonstration of the terrain and vegetation capabilities of the system, as specified by the government, has been executed. The resulting photograph is displayed in Figure 6.1.

The production steps leading to this photograph include the establishment and incorporation into the data base of the terrain library and the forest library for a task site described by the government. These steps will be discussed in Section 6.1 and 6.2, respectively. The data for image generation, shadowing and color determination will be discussed in Section 6.3.

The photograph of Figure 6.1, at a resolution of 650 x 1300 raster points, required three hours of computing time on the IBM 360/65 - well within the predicted times. The corresponding time for the CDC-6600 is about one hour. At this resolution, and seen in full color (which unfortunately is not possible in this report), the photograph is of surprisingly good quality and displays a realism beyond our expectations.

It must be concluded that the code system described in this report contains all of the technology required to produce, with reasonable manpower, and in reasonable computing times, high quality realistic photographs of large scale terrestrial scenes. The degree to which the data base and resulting photographs will represent an actual test site will be limited largely by the input description of that site.

6.1 Terrain Description

A square area, 1400 kilometers on a side, was selected by the government, and a terrain description was provided in the following form:

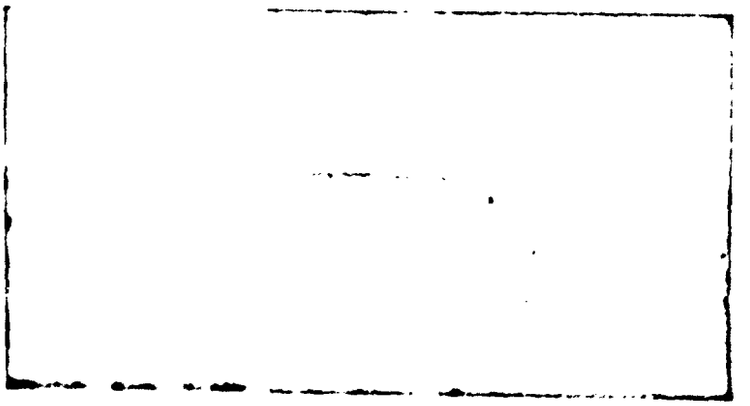


Figure 6.1 A Photograph of the Simulated Test Site

a) Elevation values, in feet, from an elevation map, for 92 points. A few of these are plotted in Figure 6.2, for reference. The rest are given in Table 6.1.

b) A few country roads, produced from memory. These are also plotted in Figure 6.2.

c) Several ground-to-ground photographs of the area, which were to provide some feeling for color boundaries and colors.

Ideally, the road descriptions and color information should come from a single overhead photograph, with known camera position and with a couple of landmarks to tie the photograph to the elevation map.

Since such a photograph was not available, color boundaries were selected by MAGI to provide a mix of sandy and grassy areas similar to that of the available ground-to-ground photographs. These boundaries are indicated in Figure 6.2 by dashed lines. It was intended that these areas be more sandy than the rest of the regions; hence several of them were drawn from points of high elevation down to points of low elevation, i.e., to behave like "wash-outs". One of these regions, namely that marked 12, was represented as a wide road, rather than as an explicit color boundary.

These data pressed the limits of the arrays and the terrain installation code and the terrain ray-tracing codes. However, the computer time consumed by the installation (PROGRAM INSTAL) was less than three minutes on the IBM 360/65, corresponding to less than a minute on the CDC 6600.

6.2 Tree Distribution

The boundaries of forested areas, several lines of trees and several individual trees were specified by the government, as shown in Figure 6.3 and Table 6.2. Two tree types were designated the pine and the oak tree - and the required means, variances and densities were provided by the government as described in Section 3 of this report.

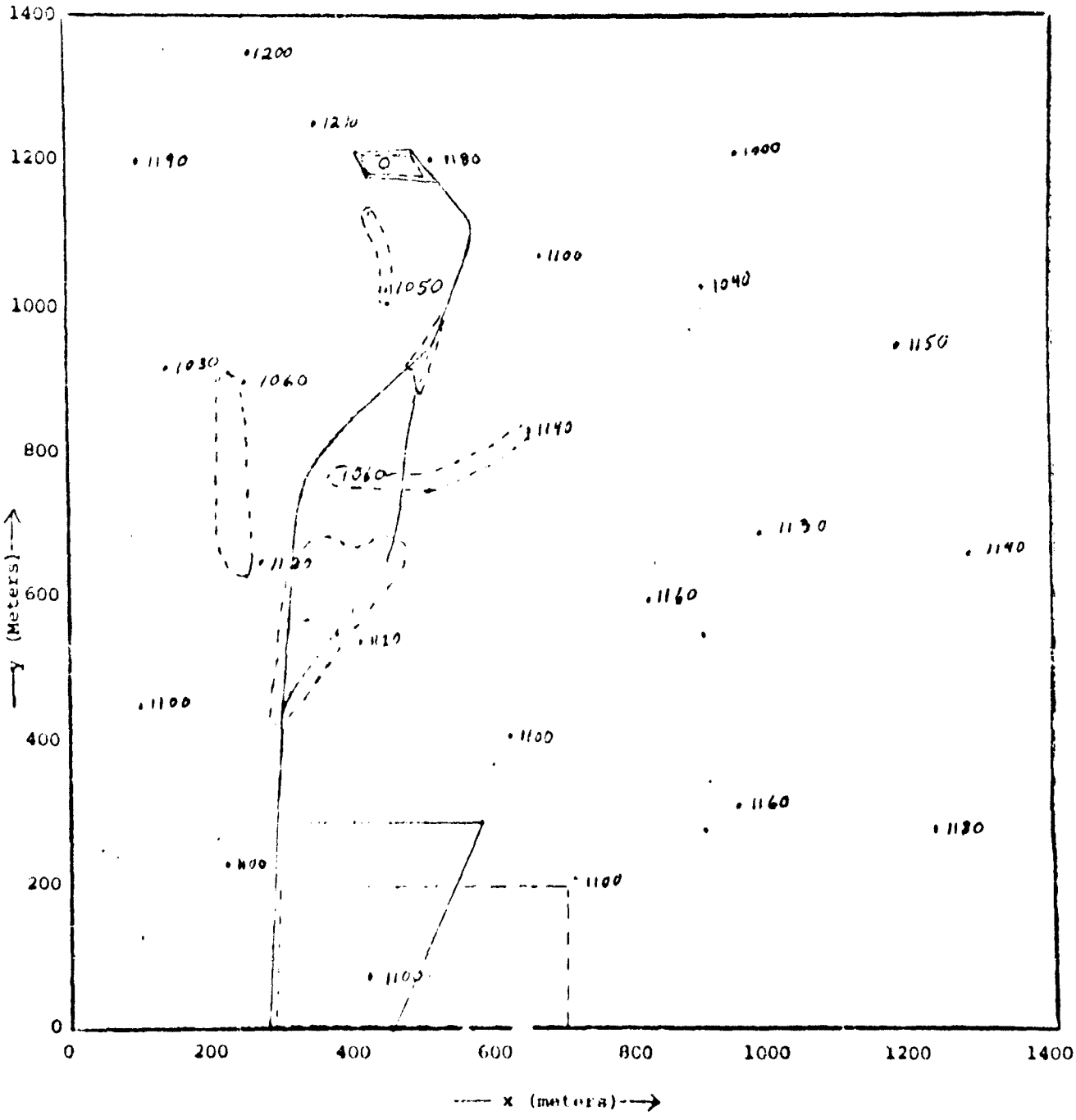


FIGURE 6.2. A Few of the Elevations (ft), the Roads (Full Lines) and Color Boundaries (Dashed Lines).

Table 6.1 Terrain Elevation Table

z_0 = 1000 feet

x (meters)	y (meters)	$z - z_0$ (feet)	x (meters)	y (meters)	$z - z_0$ (feet)
110.0	210.0	100.0	620.0	720.0	100.0
70.0	350.0	60.0	540.0	750.0	80.0
100.0	350.0	100.0	620.0	780.0	100.0
100.0	450.0	100.0	510.0	600.0	100.0
30.0	550.0	120.0	540.0	860.0	120.0
110.0	650.0	140.0	550.0	1000.0	100.0
110.0	760.0	100.0	510.0	1050.0	100.0
140.0	920.0	30.0	550.0	1200.0	180.0
30.0	950.0	100.0	510.0	1300.0	100.0
140.0	1010.0	100.0	600.0	1300.0	100.0
110.5	1060.0	160.0	770.0	110.0	100.0
100.0	1200.0	190.0	800.0	160.0	120.0
50.0	1350.0	140.0	670.0	600.0	140.0
230.0	220.0	100.0	720.0	660.0	120.0
300.0	300.0	100.0	680.0	780.0	120.0
250.0	550.0	120.0	650.0	830.0	140.0
270.0	650.0	120.0	650.0	950.0	100.0
260.0	860.0	60.0	670.0	1070.0	100.0
250.0	1040.0	100.0	730.0	1250.0	140.0
270.0	1120.0	160.0	650.0	1300.0	100.0
250.0	1250.0	180.0	920.0	80.0	150.0
260.0	1350.0	200.0	920.0	190.0	140.0
460.0	200.0	100.0	820.0	600.0	100.0
410.0	540.0	120.0	800.0	860.0	100.0
450.0	670.0	100.0	800.0	1020.0	40.0
330.0	630.0	100.0	900.0	950.0	100.0
360.0	700.0	80.0	770.0	1180.0	100.0
340.0	770.0	60.0	780.0	1290.0	100.0
390.0	850.0	100.0	1000.0	90.0	120.0
410.0	910.0	120.0	1020.0	210.0	120.0
350.0	990.0	100.0	1100.0	300.0	140.0
390.0	1060.0	100.0	950.0	360.0	160.0
450.0	1030.0	50.0	1030.0	510.0	140.0
320.0	1100.0	200.0	900.0	690.0	130.0
440.0	1220.0	200.0	950.0	1200.0	0.0
350.0	1250.0	210.0	1260.0	90.0	100.0
450.0	1350.0	160.0	1220.0	150.0	100.0
620.0	910.0	100.0	1230.0	290.0	160.0
660.0	210.0	100.0	1290.0	660.0	140.0
510.0	250.0	120.0	1120.0	750.0	140.0
520.0	400.0	100.0	1250.0	880.0	120.0
670.0	450.0	120.0	1100.0	950.0	150.0
530.0	550.0	90.0	1180.0	1010.0	100.0
570.0	570.0	100.0	1110.0	1120.0	100.0
500.0	660.0	100.0	1300.0	1140.0	100.0
620.0	650.0	120.0	1320.0	1300.0	0.0

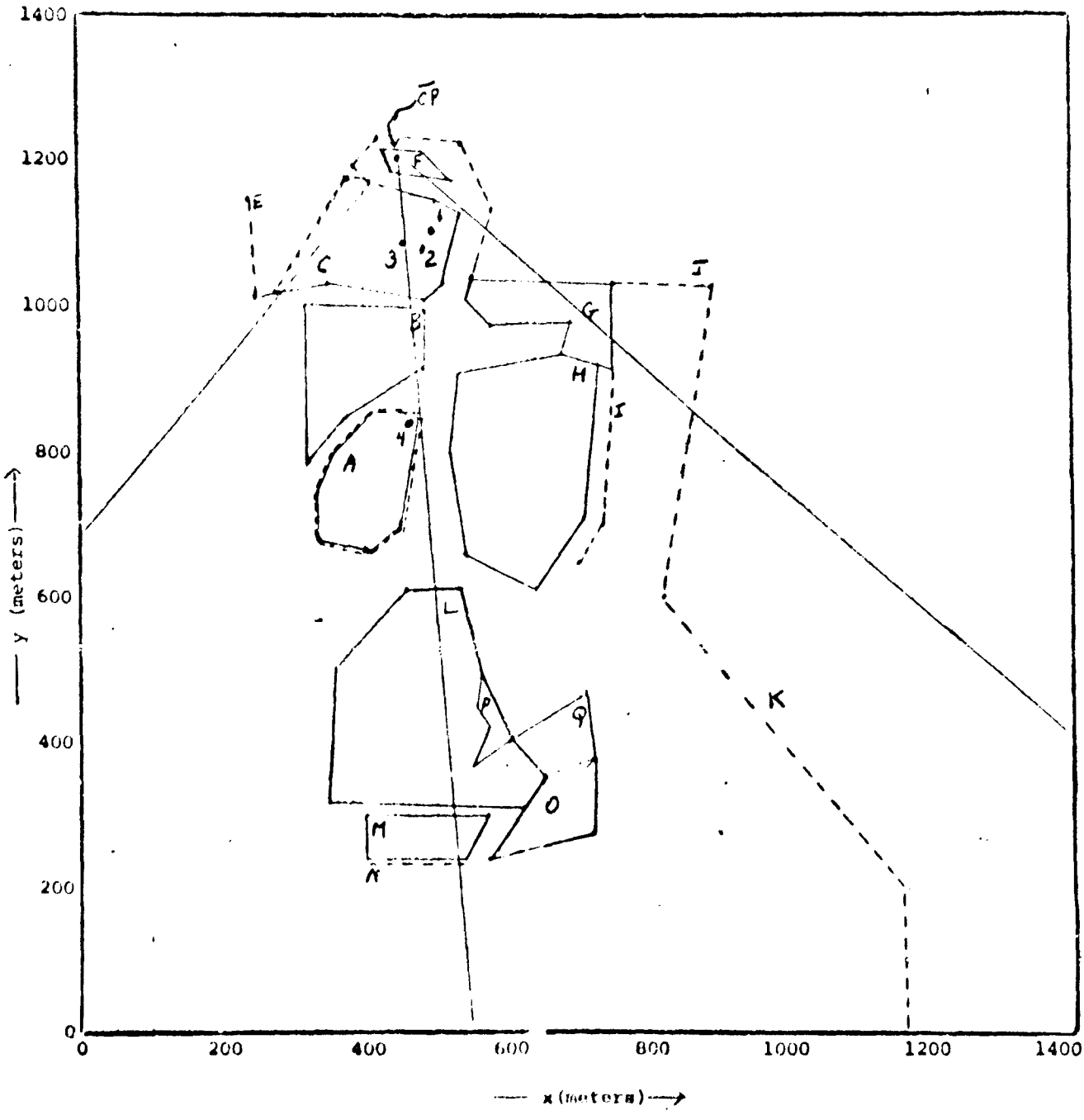


FIGURE 6.3. Boundaries of Forested Areas (Full Lines), Lines of Trees (Dashed Lines) and Individual Trees. Types, Numbers and Heights are given in Table 6.2.

Table 6.2 Tree Data For Figure 6.3

REGION	REGION TYPE	TREE TYPE	NUMBER	HEIGHT RANGE (meters)
1	Individual	21	1	17.1
2	Individual	21	1	14.8
3	Individual	21	1	13.1
4	Individual	21	1	16.4
A	Boundary	11	260	2.5 - 7.5
		21	210	9.6 -14.4
A	Line	11	113	2.8 - 5.2
A	Line	11	32	0.5 - 1.5
B	Boundary	11	82	0.5 - 1.5
C	Boundary	11	21	2.0 - 6.0
		21	108	2.5 - 7.5
E	Line	11	205	6.3 - 7.7
F	Boundary	11	24	1.3 - 1.7
G	Boundary	11	109	6.3 - 7.7
H	Boundary	11	421	7.2 - 8.8
I	Line	11	42	7.2 - 8.8
J	Line	11	13	6.3 - 7.7
K	Line	11	40	1.2 - 4.5
		21	35	1.6 - 6.0
L	Boundary	11	39	1.0 - 1.9
		21	39	1.5 - 2.5
M	Boundary	11	17	1.6 - 2.4
		21	12	2.0 - 3.0
N	Line	11	29	6.4 - 9.6
O	Boundary	11	14	7.0 -13.0
P	Boundary	11	40	4.0 - 6.0
Q	Boundary	11	52	2.8 - 5.2

Again, an overhead photograph would have served to provide these inputs for the forest generator more reliably than is possible from memory.

PROGRAM FORGEN produced something more than 2000 tree boxes in about a minute of computing time on the IBM 360/65. A printer plot (elongated) was provided by the program PLOTFR and is shown in Figure 6.4.

6.3 The Photograph

The camera position and direction were selected by the government and are shown in Figure 6.1 as $\overline{CP}$ and $\overline{CD}$, respectively. The position is near the top of a sharp rise (x=450 m, y=1200 m, z=67.5 m) and the direction is slightly below the horizontal (about 5°) and nearly coincident with the negative y axis ($\overline{CD}$ = (+ 0.1, -0.99, -0.1)). The sun was placed behind the camera with rays at an angle of 45° with the vertical and along the y-axis.

The field of view was of half-angle 22.5° in the vertical direction and half-angle 45° in the horizontal direction. Since, the required resolution (650 x 1300 raster points) exceeded the system limits of 1000 raster points in either direction, the field was divided down the middle, and two offset images (650 x 650) were generated. These were later combined by a photographic laboratory.

Table 6.3 is a listing of the input required by the image generating system to produce an image of the right hand field of view. Only one pair of numbers need by changed (in this case only one number) to produce the left hand field of view: the offset pair (0,-0.5) must be replaced by the pair (0.0, +0.5).

The total running-time for the shadowed picture has been mentioned already as three hours on the IBM 360/65. About half of this was needed for imaging vegetation, somewhat less than a quarter of it for imaging terrain and somewhat less than a quarter of the time was used in shadowing with the vegetation.

Table 6.3 Sample Input to the Image Generating System

```
ICAM        IVEG                    0
 450.0       1200.0      67.5        0.0995      -0.99       -0.1          0     0
 1.0         1.0         1.0         0.0         -0.5
  650   650
IGEO        IVEG                   14
            0           9           2           1           4
            3


    1   650     1   650
IGEO        ITER
            9           2
            4


    1   650     1   650
ICAM        IVEG                    1
 450.0       1200.0      67.5        0.0995      -0.99       -0.1          0     0
 1.0         1.0         1.0         0.0         -0.5
  650   650
            0   1.0 E+03 0.0         1.0         1.0
ISHA        IVEG                   14
            2           9                       1           4
            3


    1   650     1   650
ICON
            9          14
            4           0         0.5           0.          1.          1.
           30
   11      11      11      11      11      11      11      11      11      11
   11      11      11      11      11      11      11      11      11      11
   11      11      11      11      11      11      11      11      11      11
            5
   33      34      35      36      37
           20
   13      13      13      13      14      14      14      15      16      15
   16      17      17      17      12      12      12      12      13      13
   13      12      12      12      12      13      13      13
           35
    3       4       5       6       7       8       9      10      11      12
   13      14      15      16      17      18      19      20      21      22
   23      24      25      26      27      28      29      30      31      32
    3       4       5       6       7
IEND

EOR
```

About ten minutes were required for the conversion pass. No terrain shadows were possible with the sun at a reasonable angle, but a terrain shadow pass would have added about twenty minutes to the total.

In default of good information, colors were chosen to be pleasing and "reasonable". Use was made of the color-mixing technique[4] to give texture to the ground. All terrain regions and roads were given a reddish sand background to which varying amounts of green were added for the terrain regions, and a small amount of black was added for roads. The machine time required for a color pass is a few minutes on the IBM 360/65.

Finally, we call attention to the fact that the trees of region B (Figure 6.3) are not seen in the photograph. These fifty old pine trees ranged in height from 0.5 meter to 1.5 meters and thus, at the distance of two hundred to four hundred meters, an individual could cover no more than two or three raster points. The initial coarse grid on which rays were fired had a spacing of eight raster points and it was therefore inevitable that most or all of these small, widely spaced, specimens should be missed. Although, in this case, most of the trees were too small to be seen even if struck, a borderline case can be treated in the following way: the vertical and horizontal raster limits of the region involved should be determined and the camera option of the executive should be invoked with an initial grid spacing of 4, 2 or 1 raster points, followed by the vegetation image option with the above raster limits. This can be followed by another "camera" run with the coarser grid before going on with the other elements of the picture.

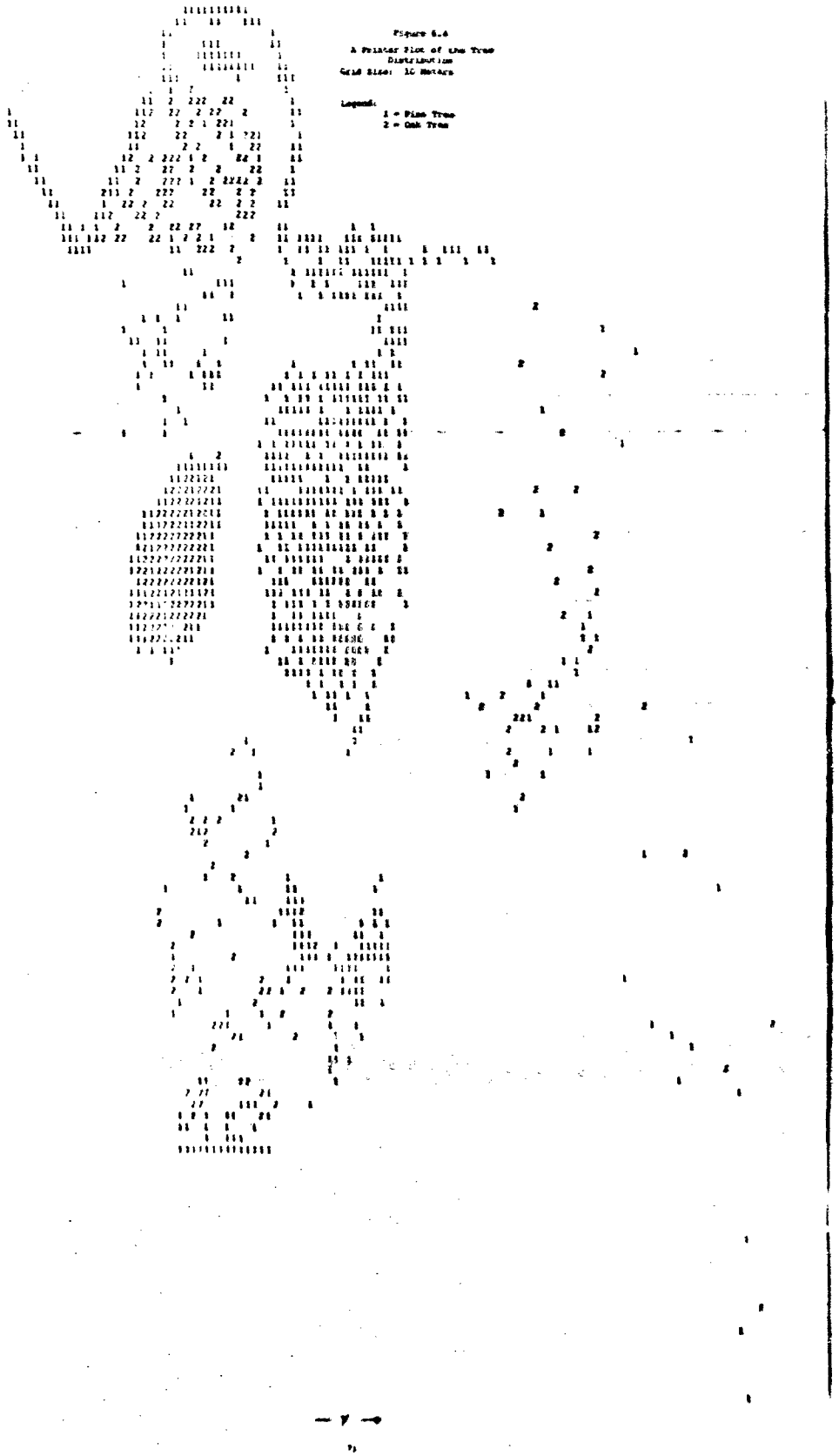


Figure 6.6
A Printer Plot of the Tree Distribution
Grid Size: 10 Meters
Legend:
1 = Pine Tree
2 = Oak Tree
N
W
E
S

7. REFERENCES

1. J. Brooks, H. Lichtenstein and H. Steinberg, "Computer Simulation of Terrestrial Scenes", BRL CR241, Mathematical Applications Group, Inc. (July 1975).

2. H. Lichtenstein, F. Rahn and H. Steinberg, "CAMNET: A Computer Simulation System for Geometrically and Optically Realistic Camouflage Nets", BRL CR160, Mathematical Applications Group, Inc. (June 1974).

3. H. Lichtenstein and H. W. Bang, "CG: A Combinatorial Geometry Module for the MICOM Lazer Terminal Homing Systems Program", MR7034, Mathematical Applications Group, Inc. (July 1973).

4. J. Brooks et. al., "An Extension of the Combinatorial Geometry Technique for Modeling Vegetation and Terrain Features", BRL CR159, Mathematical Applications Group, Inc. (June 1974).

APPENDIX A - VEGETATION ROUTINES

INTRODUCTION

In addition to the vegetation ray tracing routines in the picture making system, there are two stand-alone programs which are used to generate the tree data set for the data base. The two stand-alone programs are:

a. A Tree Construction Program which generates a high resolution data set from card input.

b. A Low Resolution Construction program which converts the high resolution data set into a low resolution data set.

These two programs and the vegetation ray tracing routines will be described here. There are many routines in the present ray tracing routines and construction programs that are identical to those routines in the previous multistage vegetation model. Details concerning these routines can be found in references 1 and 4.

THE TREE CONSTRUCTION PROGRAM

The construction program is written in six overlays. Its primary purpose is to generate a two-stage data set for an individual vegetation body. The program requires the same input as the original multi-stage model with the exception of one additional option card*. When used in the construction mode, the program generates a data file containing:

a. The location and dimensions of all the bodies (cones and leaf cloud boxes) of the tree in its own coordinate system.

b. Three dimensional sort of these bodies into a 15 x 15 x 25 cell structure.

c. The packed arrays specifying the internal structure of the lead clouds.

d. Miscellaneous ray trace data. These data are used by the high resolution ray trace program.

*For details concerning input, see documentation for the multi-stage model.

OVERLAY STRUCTURE AND FUNCTION

OVERLAY	ROUTINES	FUNCTION OF OVERLAY
(0,0)	TR DRIVE CROSS* WRITE 2 SET UP* TT 2 EQUIV* BLOCK DATA* EQUIV 2 VEGET ANGLES* CAMERA* TT READ* WRIT* BE LONG*	Overlay is primarily responsible for calling other overlays. Contains routines used by more than one overlay. Overlay also generates data (in subroutine VEGET) for multiple trees to be used by multi-stage ray tracing in overlay (5,0)
(0,1)	UNPACK REDUCE	Uses data generated by overlay (3,0) to create arrays containing the location and position of all the bodies in tree. Routines use the multi-stage data and produce the required two stage data (TRC data and leaf cloud box data)
(2,0)	SET UP 2 UNPACK PACKER INCLUB SQUARE B BELONG C	Uses data generated by overlay (1,0) to generate three dimensional cell structure. Cell structure data specifies which of the many bodies are contained in each of the cells covering the tree

OVERLAY STRUCTURE AND FUNCTION

OVERLAY	ROUTINES	FUNCTION OF OVERLAY
(3,0)	TREE BDY DAT COPY REGTRA	Reads data describing tree. Generates original multi-stage tree data. Data required is identical to original multi-stage data and is read from input
(4,0)	WRITER	Prints out all two stage data generated by overlay (1,0) and overlay (2,0)
(5,0)	PICTUR CLOUD FOG GO GETEC GMTRG GNOOM ROTNOR GZEXT GZINT MORK PINE RAYTRK RIGID IFILL INDEX	Contains all the routines necessary for multi-stage ray tracing. Uses data from overlay (3,0) and overlay (0,0) to produce block format image tape.
(6,0)	DATSET	Uses data from overlay (0,0) overlay (1,0) and overlay (2,0) to output complete two stage cell structured binary data set.

SEQUENCING OF ROUTINES AND DATA FLOW

The labeled diagrams illustrate the calling sequence for overlays (0,0), (1,0), (2,0), (4,0) and (6,0) and the files used by the program. Overlay (3,0) consists only of routines responsible for the construction of the multi-stage tree. The linkage and data are unchanged and will not be diagrammed. Similarly overlay (5,0) contains only multi-stage ray trace routines and will not be diagrammed. Further details concerning the linkage can be found in earlier documentation.

Note that in the diagrams some of the data is described as data (a), (b), (c), or (d). These refer to the two stage data as follows:

(a) location and dimensions of all bodies

(b) cell structure arrays

(c) internal structure of leaf clouds

(d) miscellaneous ray trace data

DESCRIPTION AND FUNCTIONS OF NEW ROUTINES

Routines marked with an asterisk in overlay (0,0) as well as all routines in overlay (3,0) and (5,0) are taken from the original multi-stage model. They are unchanged and will not be described. Flow charts are provided for selected routines.

PROGRAM TRDRIVE

TRDRIVE is main driver program. Its only functions are reading the option card, opening the files, calling the starting routines, and writing out some statistics.

SUBROUTINE WRITE2

WRITE2 writes a binary file on tape 7 containing contents of common blocks, FG, COLR, KD, KLOUD and RAYTR. These blocks contain ray tracing information which are necessary for both the original multi-stage ray trace and new two-stage high resolution RAYTRACE.

SUBROUTINES TT (IN) AND TT2 (IN, IM)

These routines simply compute processor time between various program steps and print out the time with an indicating message.

VARIABLES

IN - Alphanumeric message to be printed

IM - Numeric information in message

SUBROUTINE EQUIV2 (V, H, G, RB, RT)

Computes body data for smallest RPP which would completely contain a given cone.

VARIABLES

V(3) - cone vertex vector

H(3) - cone height vector

G(2,3) - RPP data

RB - cone base radius

RT - cone top radius

SUBROUTINE ZERIT

ZERIT is essentially unchanged, however, it does call various overlays for call structure construction based upon input option parameters.

PROGRAM UNPACK

UNPACK is the main program of overlay (1,0). Data describing the tree is stored in common blocks GEOM2, DTREE, and GEOM4 which were generated by overlay (3,0). The data is stored in the multi-stage format. UNPACK changes the data into a two stage format.

In the multi-stage format there are three types of bodies: cones, RPPs, (leaf cloud), and boxes (containing the entirety of a lower stage). UNPACK computes the rotation matrix ($\overleftrightarrow{R}$), translation vector ($\vec{V}$), and scale factor (D) that would transform body data for a lower stage body into the proper "real world" body data (complete tree reference frame).

This transformation data is computed from the stored "copy" data in the multi-stage arrays. The resultant two stage data which is produced consists of the body data for each individual cone or box (leaf cloud container) expressed in this single complete tree reference frame. The internal structure of the leaf cloud RPP is unchanged. The entire tree is enclosed in an RPP with one vertex located at the coordinate origin with the remainder of the RPP situated entirely in the positive octant.

See flowchart #1

SUBROUTINE REDUCE (SIZE, NIR)

In UNPACK, data for each body is computed relative to a complete tree reference frame which is in reality stage 5 of the multi-stage model. However, due to the construction of boxes over each 4th stage body (done in overlay (3,0)), the covering RPP is larger than is necessary. REDUCE considers all of the bodies in the two stage description and computes parameters of the smallest RPP which would completely contain all of the bodies. A translation of coordinates is made so that the RPP covering the bodies will still maintain the property of having one vertex located at the coordinate origin.

VARIABLES

SIZE (3) - SIZE (1) and SIZE (2) are x and y dimensions of the RPP divided by 15. SIZE (3) is z dimension of RPP divided by 25.

NIR - Total number of bodies in two stage description.

PROGRAM SETUP 2

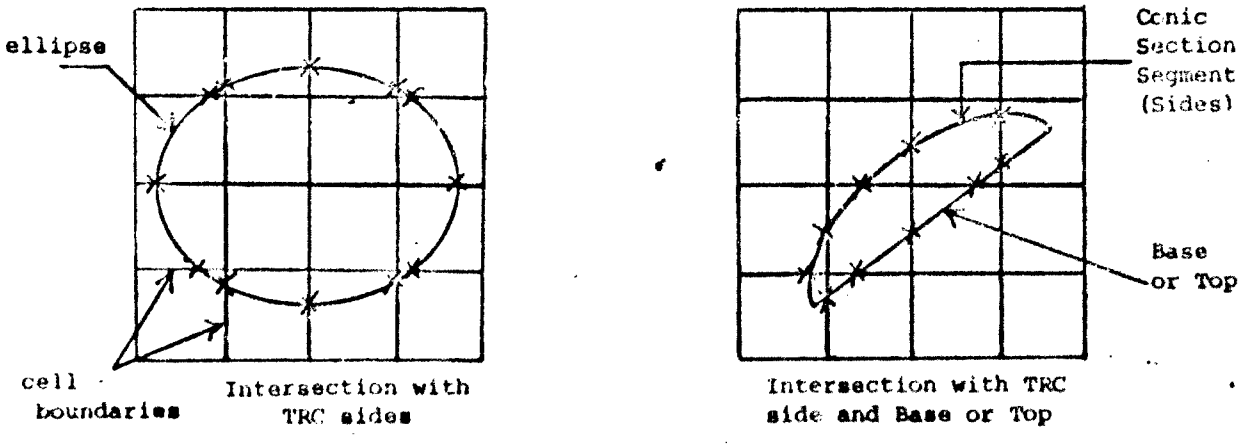
SETUP 2 is the main program of overlay (2,0). The RPP covering the tree is subdivided into three dimensional cells so that the entire RPP is covered by a 15 x 15 x 25 matrix. The body data for each individual cone or box is read from an intermediate file (TAPE 4) one by one. This routine (through its subroutines) determines which cells of the matrix contain any portion of the external surface of a given body. It thereby generates an array specifying which bodies are contained within each cell of the covering matrix. Note: if any cell is completely contained within a given body, the cell is deemed not to contain the body (i.e., only external body surfaces are considered). The routine also packs the cell content information into the words of the cell structure array.

See flowchart #2

SUBROUTINE BELONG C (XL, IR, V, H, RB, RT, CELL)

BELONG C computes which cells of the covering cell matrix contain any portion of the external surface of a given truncated right cone (TRC). First, the smallest RPP which would completely inclose the TRC is constructed. From this RPP, one determines the maximum and minimum cell indices for cells which could have a common volume with the TRC. Cell boundaries are represented by planes of constant x, y, or z. The intersection of the cones with each of the possible cell wall planes is then computed. The locus of these intersections will either be an ellipse or some portion of a conic section connected by straight lines as shown below:

CELL WALL PLANES



within these cell wall planes, there are further intersections with the two dimensional cell boundary lattice and the TRC loci as shown on the previous diagram. All cells whose boundary contains one of these point intersections deemed to contain the TRC.

The routine also packs the words for the appropriate entries in the cell structure array to reflect that the particular cell contains the given TRC (body number JR).

VARIABLES

XL (3) - dimensions of cell

IR - body number

V(3) - vertex vector for TRC

H(3) - height vector for TRC

RB - radius of TRC base

RT - radius of TRC top

CELL (15, 15, 25) - cell structure array

SUBROUTINE INCLUD (GM, GL)

Routine is called by BELONG C. Variables GM and GL correspond to values for a point intersection between a TRC and two dimensional cell boundary lattice on a given cell wall plane. INCLUD computes the indices of the cells whose boundary contains the point intersection.

SUBROUTINE BTINT (LBT, RB, CB, A, B, GK, K, L, M, GLO, GLI, GMO, GMI)

Routine is called by BELONG C. Its function is to determine whether the TRC base or TRC top intersect with a given cell wall plane. If either the base or top does intersect, BTINT also computes the endpoints of the line of intersection.

VARIABLES

IBT - Logical variables which is true when intersection occurs.

RB - radius of TRC top or base (whichever is being considered)

CB - equals either 0 or 1.0 0 → base 1.0 → top

A(3,3) - elements of a matrix satisfying $\sum_{ij} A_{ij}X_iX_j = R(t)$ $0 \leq t \leq 1$ which represents outer surface of cone sides. Note t = 0 → base t = 1.0 → top

B(3) - coefficients describing base or top plane.

GK, K, L, M - set of parameters describing cell wall plane.

K = 1, L = 2, M = 3 constant x plane at x = GK

K = 2, L = 3, M = 1 constant y plane at y = GK

K = 3, L = 1, M = 2 constant z plane at z = GK

(GLO, GMO); (GLI, GMI) - two end points of intersecting line

SUBROUTINE BELONG B (V, H1, H2, H3, CELL, NCELL, ENDS, ML, XL)

BELONG B computes which cells of the covering cell matrix contain any portion of the external surface of a given box (leaf cloud container). Again the boundaries of cells are represented by planes of constant x, y, or z. The intersection of any box face with these planes are line segments. The routine computes these line segments (including their end points) for each of the possible cell wall planes. Point intersections of the remaining cell wall boundaries (the two dimensional lattice of cell boundaries in each cell wall plane) with the box plane line segments are then computed. Any cell whose boundary contains these point intersections are deemed to contain the box in question.

SUBROUTINE INCLUB (X, Y, Z, III, JJJ, CELL, XL)

Routine is called from BELONG B. Variables (x, y, z) correspond to values for a point intersection between a box face and the two dimensional cell boundary lattice on a given cell wall plane. INCLUB computes the indices of the cell whose boundary contains the point intersection.

SUBROUTINE PACKER (K, I, J, IR)

If more than 4 bodies are contained in any cell, it will require more than one word to list the body numbers. PACKER temporarily stores these body numbers on random mass storage (RMS) disk.

VARIABLES

K - cell number $1 \leq K \leq 5625$

I - present contents of cell array

J - number of bodies in cell

IR- body number

SUBROUTINE REPACK

For all cells containing more than 4 bodies, body number is stored on RMS. After all bodies have been considered, subroutine REPACK recalls the data and packs the words of the cell array (Cell (15,15,25)) and packs the overflow data in an addition array (Cell 2).

PROGRAM DATSET

Routine reads information from three files (TAPE3, TAPE4, and TAPE7). These files all contain different portions of the total tree data set. DATSET then writes the entire data set on one file TAPE3.

REQUIRED INPUT

Program requires the same input as original vegetation construction program with the exception of the first card (option card).

Option Card (ISELEC(I), I=1,3)

Format (3(I2, IX))

Where

ISELEC (1) =	-1	No cell structured data set
	1	Create cell structured data set
	3	No printout of cell structured data
ISELEC (2) =	0	No ray trace
	1	Ray trace with multi-stage data set
ISELEC (3) =	1	No ray trace and no printout of multi-stage data set
	0	Printout multi-stage data set

FORMAT OF DATA SET

The output file (TAPE3) contains two files each of one record and all in binary format.

File 1

There is a total of 21631 words on file 1 in the following arrays:

MA (3000)	Contains integer body data for body number I
MA (2*I-1)	Contains body type (5-TRC; 7-BOX)
MA (2*I)	Contains pointer to location in FPD array containing real body data
MA (3001-I)	Contains region number
FPD (7500)	Contains real body data
For TRC -	8 words (3 for vertex vector; 3 for height vector; Base radius; and top radius)
For BOX -	12 words (3 for vertex vector; and 1 for each of the 3 height vectors

CELL (15,15,25) - Contains body numbers for those bodies in given element of the three dimensional lattice. All words are packed integers. This is called the cell structure array.

If there are 1 or 2 bodies in the given cell, word is packed as follows:

$$2^{16}*(2^{nd} \text{ body \#}) + 2^{6}*(1^{st} \text{ body \#}) + (\text{\# of bodies in cell})$$

If there are greater than 2 bodies in given cell, word is similarly packed except that the second body # is replaced by pointer to location in CELL2 array where remainder of body numbers can be found.

CELL2 (3000) - Contains packed integers with body numbers for those cells containing more than two bodies. There are 3 additional body numbers in each word. Additional body numbers for the given cell are found as necessary in succeeding words of the array.

Integer word is $\sum_{i=1}^{30} (\text{Body \#})\ 1024^{(i-1)}$

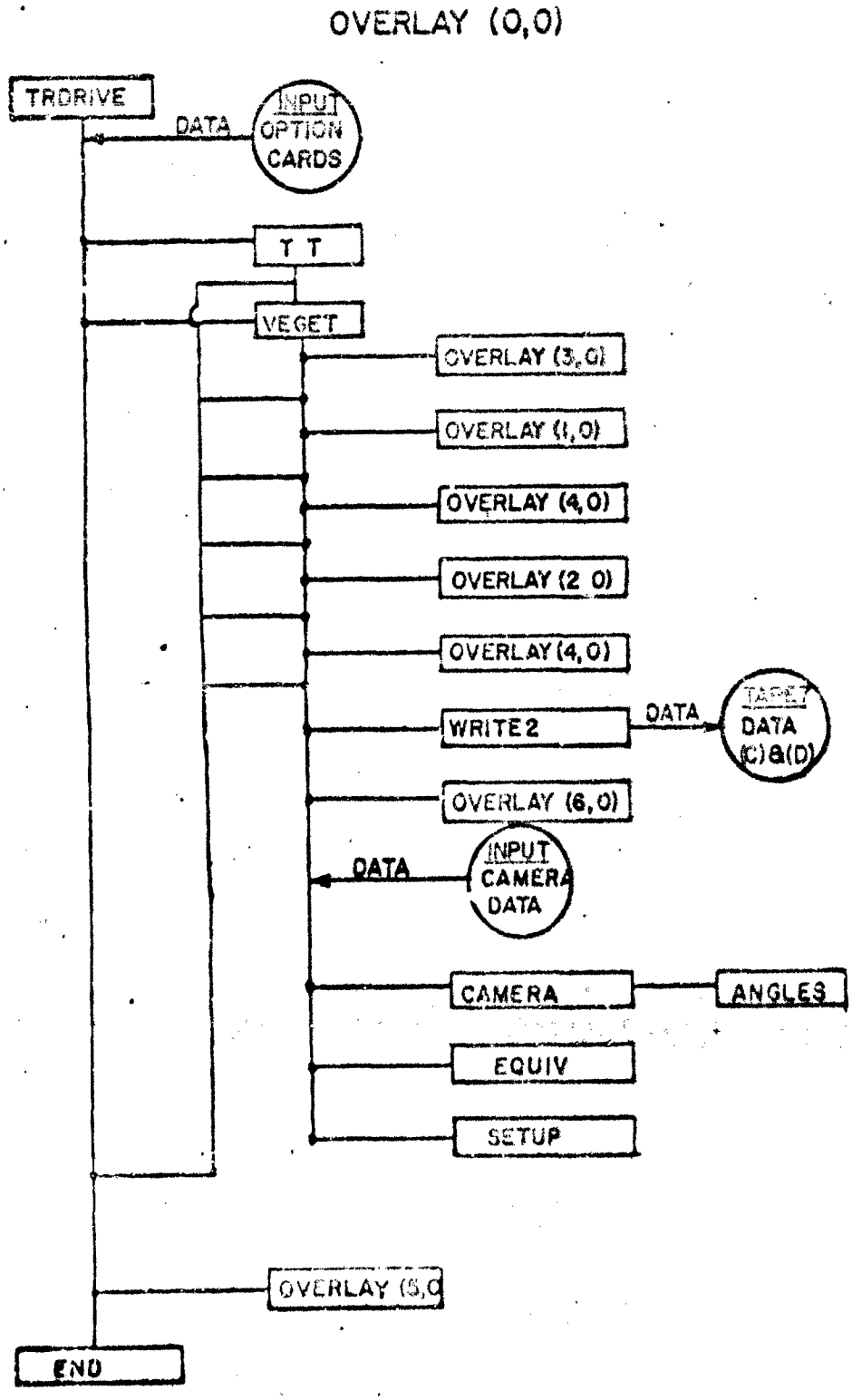
XLEAF	-	X dimension of 1st stage leaf cloud RPP
NTRC	-	Location of Body data for 1st stage cone in RPD array
XL(3)	-	Dimensions of a unit cell
NLEAF	-	Location of RPP data for 1st stage leaf cloud in RPD array.

File 2

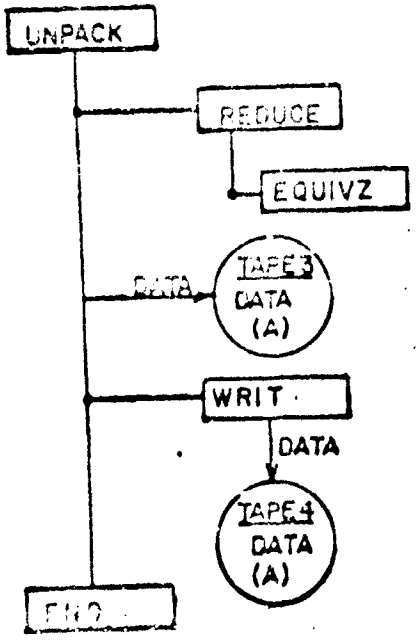
This file has 1671 words and contains the contents of the following common blocks:

/PV/ ; /CELL/ ; /RD/ ; /CLOUD/ ; and /PAYTR/

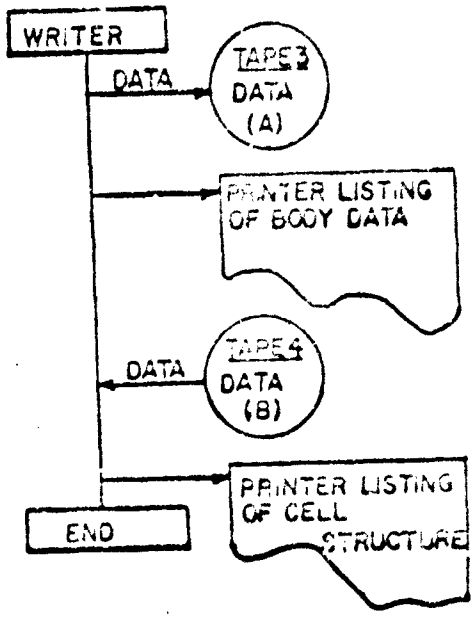
These common blocks contain miscellaneous data used by the ray trace routines. The structure and contents of these common blocks are identical to those in the original multi-stage model.


OVERLAY (0,0)
TRDRIVE
DATA
INPUT OPTION CARDS
T T
VEGET
OVERLAY (3,0)
OVERLAY (1,0)
OVERLAY (4,0)
OVERLAY (2 0)
OVERLAY (4,0)
WRITE 2
DATA
TAPE 7 DATA (C) & (D)
OVERLAY (6,0)
DATA
INPUT CAMERA DATA
CAMERA
ANGLES
EQUIV
SETUP
OVERLAY (5,0
END

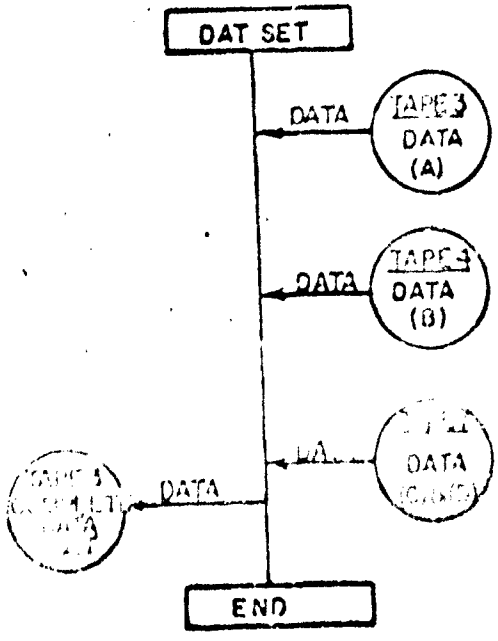
OVERLAY (1,0)



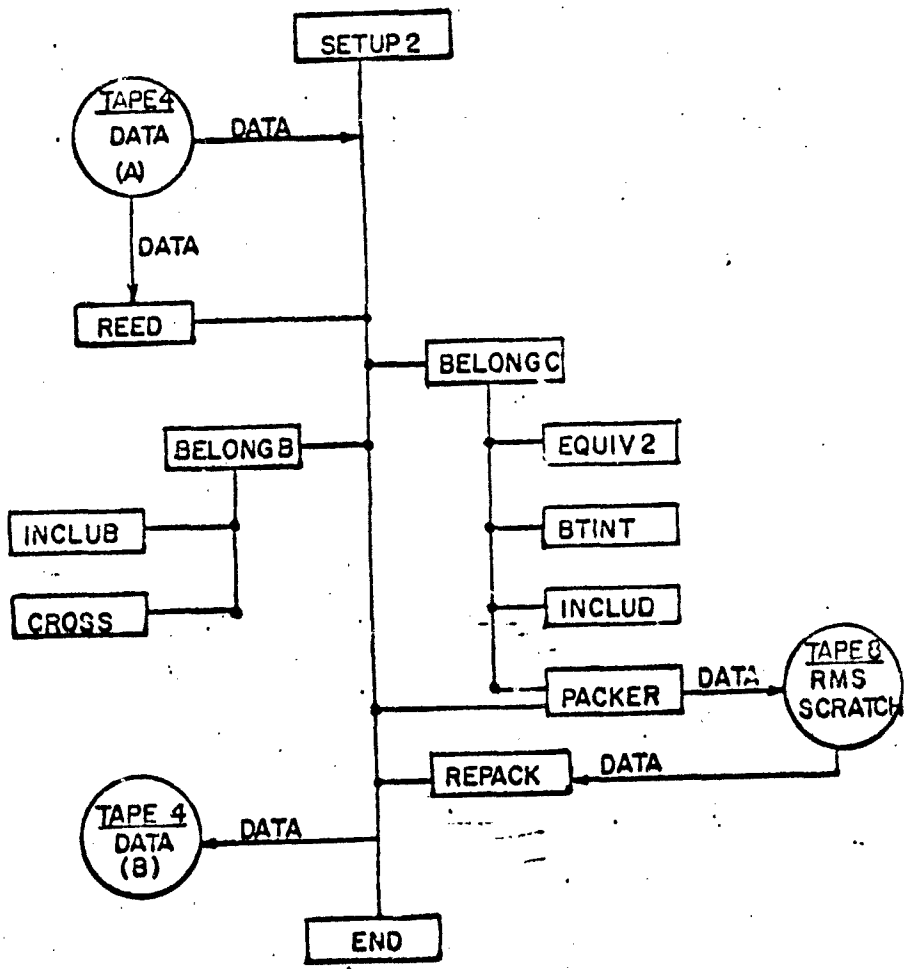
OVERLAY (4,0)



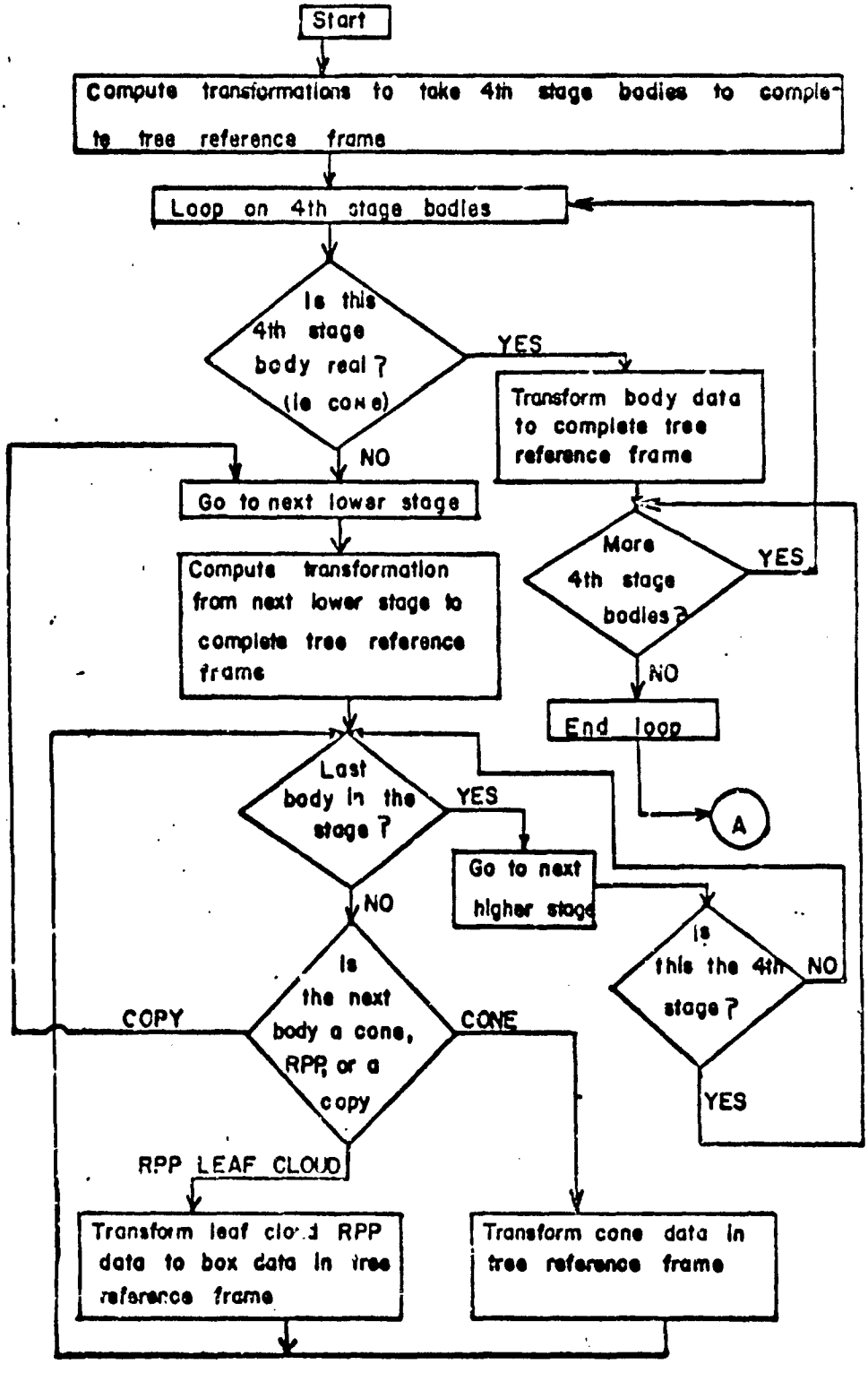
OVERLAY (6,0)



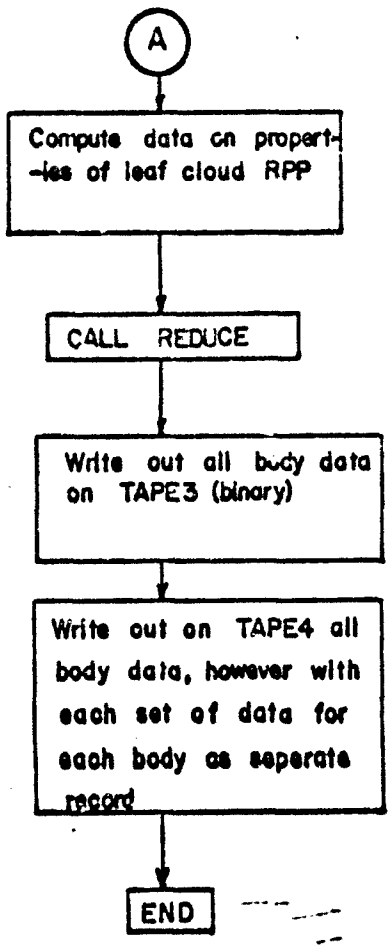
OVERLAY (2,0)


SETUP 2
TAPE 4
DATA
(A)
DATA
DATA
REED
BELONG C
BELONG B
EQUIV 2
INCLUB
BTINT
CROSS
INCLUD
PACKER
DATA
TAPE 8
RMS
SCRATCH
REPACK
DATA
TAPE 4
DATA
(B)
DATA
END

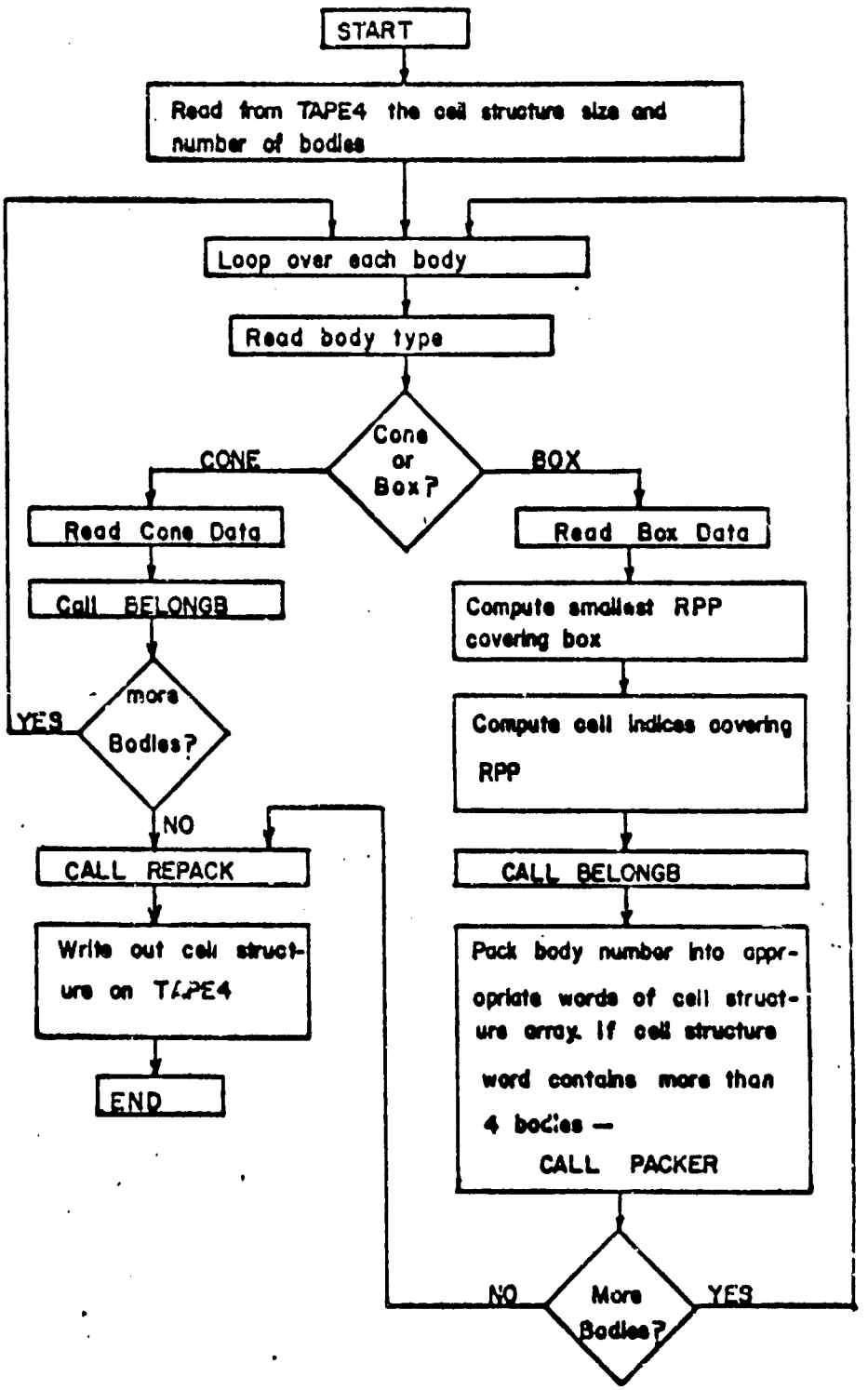
FLOWCHART #1 - UNPACK



FLOWCHART # 1 UNPACK (Cont.)



FLOWCHART #2 - SETUP2



LOW RESOLUTION CONSTRUCTION PROGRAM

The low resolution construction program is a stand-alone program which converts a high resolution data set into a low resolution data set. This data set has the following information:

a. The location and dimensions of all large TRCs making up the trunk and major branch structure of the tree.

b. Three dimensional cell structure description (consists of sort of large TRCs and mean free path information for the 15 x 15 x 25 cell structure).

c. Miscellaneous ray tracing data

The program reads the high resolution data set and determines which of the TRCs are to be retained as real bodies (basically all TRCs are retained except the 1st stage TRCs from the old multi-stage model corresponding to secondary branches).

The program reduces the size of the body data arrays since box data for leaf cloud containers and 1st stage TRCs are no longer stored.

A new cell structure is computed. This array contains body numbers for all of the remaining TRCs. In addition, this array contains mean free path information for rays traversing the cells. This information is computed by considering how many TRCs or leaf cloud boxes have common volumes with the cell in question.

VEGETATION RAY TRACING

High Resolution

The high resolution ray tracing produces a block format image tape identical to the image tape produced by the multi-stage model. The following routines are responsible for the high resolution ray tracing:

PICTUR**	RAYTR2	MURK*
INDEX**	RIGID	PINE*
RAYTRK	NEWGG	CLOUD*
RAYTR1	FOG*	

The routines marked with an asterisk are unchanged from the original multi-stage model. Those marked with a double asterisk are common to other types of ray tracing (i.e., conventional geometry, terrain geometry, or camouflage net geometry). These routines will not be described.

SUBROUTINE RAYTRK (CP, CD, NC)

This routine is responsible for assuring that the proper trees are in memory for the ray tracing. Due to the large number of trees possible in the scene, it was necessary to have pre-sorted which trees could be struck by rays from a given block on the block format image tape. This sorted information is on a disk file (TAPE 10). In addition geometry information (box data) for only a limited number of trees can be in memory at any given time. Hence, the box data for the trees must be read into memory in groups of 200. This box data is stored on disk file TAPE 3. RAYTRK reads the file TAPE 10 and determines to which of the 200 resident trees ray tracing should be attempted on the basis of the block indices for the current ray. RAYTRK will also read in another group of 200 trees from TAPE 3 as required.

Called from: PICTUR

Variables: CP (3) Camera position coordinates
CD (3) Ray direction
NC Region strike number

Routines called: RAYTR1

SUBROUTINE RAYTR1 (CP, CD, NC)

RAYTR1 is responsible for ray tracing to the boxes containing trees. If a tree box is struck, the ray is transformed into the reference frame of the tree and internal ray tracing is performed by RAYTR2. If an internal ray strike is made, RAYTR1 transforms the normals back into the real world reference frame.

Called from: RAYTRK

Variables: CP (3) Camera position coordinates
CD (3) Ray direction
NC Region strike number

Routines called: RAYTR2, NEWGG

SUBROUTINE RAYTR2 (DIS, CP, CD, NC, IMISS)

This routine is responsible for ray tracing inside a tree box. A transformed ray is followed through the cell structure. Tracing is done to the geometric solids which comprise the tree. The trunk and limb structure is composed of Truncated Right Cones (TRC) and the leaf and twig structure is contained in boxes. If a ray strikes a cone, a normal is computed and ray tracing ends for the ray. If a leaf cloud box is struck, the ray is again transformed into the reference frame of the leaf cloud containers and internal leaf cloud tracing is done by the subroutines FOG, CLOUD, MURK, and PINE. Due to the great number of bodies (TRC, and leaf cloud boxes) comprising the tree, there is a covering three dimensional cell matrix over the entire tree. The path of the ray intersects only a few of the cells of the matrix and tracing is done only for those bodies along the path of the ray.

Called from: RAYTR1

Variables: DIS - distance to ray strike from tree box surface
CP (3) - coordinates on tree box surface where ray enters
CD (3) - ray direction in tree box frame
NC - region strike number
IMISS - logical variable which is true of ray misses all internal tree bodies

Subroutines called: RIGID, NEWGG, FOG

SUBROUTINE RIGID (IBOX, EXIT, START, CLAST, CD, NN)

Subroutine RIGID uses the cell structure array to determine which bodies are contained within any given cell of the covering three dimensional matrix. The body numbers are returned to RAYTR2 for ray tracing.

Called from: RAYTR2

Variables:

IBOX (64)	-	array containing body numbers of those solids (TRCs or leaf cloud boxes) to be traced toward
EXIT	-	logical variable which is true if ray has exited the tree box
START	-	distance ray has traveled point of entry into tree box
CLAST (3)	-	coordinates on tree surface where ray enters
CD (3)	-	ray direction in tree box frame
NN	-	number of body numbers in array IBOX

Subroutines called: None

SUBROUTINE NEWGG (ISIDE, DIS, BOX, ITYPE, CD, CP, IMISS)

This routine computes the strike distance for intersections of rays with the three dimensional solids (TRCs or boxes).

Called from: RAYTR1, RAYTR2

Variables:

ISIDE	integer specifying which surface strike occurs (i.e., flat surface or cone side)
DIS	distance from ray origin to strike point
BOX (12)	array containing body data (i.e., box height vectors or TRC data)
ITYPE	integer specifying body type (TRC or box)
CD (3)	ray direction
CP (3)	coordinates of ray origin
IMISS	logical variable which is true of ray misses body

Routines called: None

Low Resolution

In the high resolution model, all ray strikes are computed as intersections with three dimensional solids or as strikes with leaves or twigs inside leaf cloud containers. In the low resolution model, ray strikes are determined probabilistically for leaf clouds and the smaller cones representing small limbs. For the trunk and major branch structure, the ray tracing is done as it is in the high resolution model. In the cell structure array (the three dimensional matrix covering the tree), mean free paths for rays traversing the cell are stored instead of body numbers for leaf cloud boxes or small cones. The body numbers for the trunk and major branch structure are also stored in the cell structure array.

The routines which are changed in the low resolution model are RAYTR2 and RIGID. In addition, the routines FOG, MURK, PINE, and CLOUD are no longer used since there is no ray tracing internal to the leaf cloud containers.

SUBROUTINE RAYTR2 (DIS, CP, CD, NC, IMISS)

The routine arguments and definition of these arguments are the same as for the high resolution RAYTR2. The low resolution RAYTR2 traces only to those TRCs comprising the trunk and major branch structure. Leaf strikes and small twig strikes are computed probabilistically. As the ray is traversing the three dimensional cell structure, the path length of the ray in each of the intersected cells is computed. Stored in the cell structure array are mean free paths (λ) for rays in that given cell. A random number is drawn (P) between 0. and 1. then a distance $R = -\lambda \ln(1-P)$ is computed. This is the distance a ray would travel in a medium of mean free path λ with a probability P.

If this distance is less than the path length of the ray in the given cell, then a ray strike is determined to be made at distance R. Otherwise the ray is determined not to have struck any leaves and continues on to the next cell provided no TRC hits are made. There is never any ray tracing either to leaf cloud boxes or internal to these boxes.

Called from: RAYTR1

Variables: Same as for high resolution RAYTR2

Routines called: RIGID, NEWGG

SUBROUTINE RIGID (IBOX, EXIT, START, CLAST, CD, NN)

This routine extracts both the mean free paths and body numbers for the large TRCs from the cell structure array for each of the cells along the path of the ray. This information is returned to RAYTR2 for processing.

Called from: RAYTR2

Variables: - Same as for high resolution RIGID

Routines called: None